

DTIC FILE COPY

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD-A196 123

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/CI/NR 88-38	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ROTC CADET INFORMATION SYSTEM		5. TYPE OF REPORT & PERIOD COVERED MS THESIS
6. AUTHOR(s) CARTER L. FRANK		6. PERFORMING ORG. REPORT NUMBER
7. PERFORMING ORGANIZATION NAME AND ADDRESS AFIT STUDENT AT: UNIVERSITY OF ARIZONA		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
12. REPORT DATE 1988		13. NUMBER OF PAGES 244
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) AFIT/NR Wright-Patterson AFB OH 45433-6583		15. SECURITY CLASS. (of this report) UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE		
16. DISTRIBUTION STATEMENT (of this Report) DISTRIBUTED UNLIMITED: APPROVED FOR PUBLIC RELEASE		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) SAME AS REPORT		
18. SUPPLEMENTARY NOTES Approved for Public Release: IAW AFR 190-1 LYNN E. WOLAVER Dean for Research and Professional Development Air Force Institute of Technology Wright-Patterson AFB OH 45433-6583 18 July 88		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) ATTACHED		

DTIC  
ELECTE  
AUG 03 1988  
S  
C/D  
D

ROTC CADET INFORMATION SYSTEM

( RCIS )

BY

Carter L. Frank

A Report Submitted in Partial Fulfillment of the  
Requirements for the Degree of Master of Science  
(Management Information Systems)  
in The University of Arizona

1987

Master Committee:  
Dr. Sudha Ram

## ABSTRACT

### ROTC CADET INFORMATION SYSTEM

by

Carter L. Frank

The ROTC CADET INFORMATION SYSTEM (RCIS) is a computerized database system that was custom developed for the U.S. Air Force AFROTC Detachment 020. RCIS assists the administrative staff by providing them with fast on-line access, for cadet file updates, for processing ad hoc cadet file queries, and for producing hardcopy reports. RCIS assists the executive staff by providing fast on-line access to essential cadet information. RCIS will be reviewed by AFROTC Headquarters for possible nation-wide implementation.

*Submitted, for review - 11/1/79*  
*Frank, C. L.*



ii

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution	
Availability Codes	
Dist	Availability Codes
A-1	

TABLE OF CONTENTS

Title Page.....	i
Abstract.....	ii
Table of Contents.....	iii

SYSTEM ANALYSIS AND DEVELOPMENT JOURNAL

USER MANUAL

TECHNICAL MANUAL



# SYSTEMS ANALYSIS AND DEVELOPMENT JOURNAL

## TABLE OF CONTENTS

Title page.....	i
Table of Contents.....	ii
1. INTRODUCTION.....	1
Name of Project.....	1
Name of Organization.....	1
Contact Person.....	1
Advisors to the Project.....	1
Short Description of the Organization.....	1
Statement of the Problem.....	2
2. DEVELOPMENT OF RCIS.....	3
Purpose of the Project.....	3
Contents of the Project.....	3
Classification of System Components.....	3
Methodology.....	4
Software Required.....	4
Hardware Required.....	4
User Documentation.....	4
3. TASK PROGRESS.....	5
Analysis of Previous System.....	5
User Review of Proposed System.....	6
Design of the Database.....	6
Design of Data Entry Utilities.....	7
Validation of Data Entry Utilities.....	7
Redesign of Data Entry Utilities.....	8
Design of Query Interface.....	8
Validation of Query Interface.....	9
Redesign of Query Interface.....	9
User Training.....	9
Installation of Final System.....	10
4. FINAL REMARKS.....	11
Contribution to the Field of MIS.....	11
Practical Experience Gained.....	11

# 1. INTRODUCTION

## Name of Project

-----

ROTC Cadet Information System (RCIS)

## Name of Organization

-----

U.S. Air Force  
AFROTC Detachment 020  
University of Arizona  
Tucson, Arizona 85721  
(602) 621-3521

## Contact Person

-----

Ms. Peggy Mittendorf  
AFROTC Detachment 020 Secretary

## Advisors to the Project

-----

Dr. Sudha Ram  
Department of Management Information Systems

## Short Description of the Organization

-----

The AFROTC Detachment 020 staff is comprised of five officers, three enlisted personnel and two civilians. This staff is responsible for the training of approximately 200 cadets as well as maintaining their records and serving as liaison for approximately 50 AFIT students attending the University of Arizona. Colonel Charlie Hastings is the detachment commander, and Ms. P. Mittendorf is his secretary. The officers reporting to Colonel Hastings are Major R. Youmans (Commandant of Cadets/Freshman Instructor), Major D.

Smith (Education & Training Officer/Senior Instructor), Captain J. Dougherty (Recruiting Officer/Sophomore Instructor), Captain K. Nonaka (Drill Team Advisor/Junior Instructor). Also reporting to Colonel Hastings is Technical Sergeant G. Cobo (Detachment Non-commissioned officer in charge). He supervises Technical Sergeant R. Nicholson (underclassmen records administrator) and Sergeant D. McGrath (upperclassmen records administrator). Mr. R. Haney serves as the Uniform Custodian and is responsible to Colonel Hastings.

#### Statement of the Problem

-----

The Detachment's present system involves a myriad of forms dealing with a variety of personnel information and suspense dates for required reports. This information includes testing, rating, and grading results as well as personal information on all of cadets and AFIT students. All information gathering is done manually.

One of the major problems the detachment staff faces under the current system is the amount of time it takes to collate information from the various Air Force and Detachment forms and present information in a usable format. This takes a considerable amount of time for one cadet and even longer when information must be gathered on different groupings of cadets.

## 2. DEVELOPMENT OF RCIS

### Purpose of the Project

-----

The purpose of the project is to develop a computerized database system (RCIS) for the AFROTC Detachment 020 Cadet files. After the database records have been fully audited, the database will provide the means for the detachment staff to quickly access their files, efficiently process a wide variety of ad hoc queries, and produce hard-copy reports based on those queries.

### Contents of the Project

-----

RCIS will provide a menu-driven interface that allows the user to enter, update, archive and delete cadet records that are now stored on various forms in large metal filing cabinets. The system will also provide a query interface that allows the user to develop an ad hoc query without using the dBASE III PLUS command language. Finally, RCIS will provide utilities that automatically create backup copies of required system database files.

### Classification of System Components

-----

The system contains the following:

- a. Assembly language driver to create pop-up menus  
(Created by Stephen M. Curran)
- b. Data entry and update screens
- c. Review screens
- d. Ad hoc query generator
- e. Archive utilities
- f. Automatic backup and reload utilities

## Methodology

-----

The following methodology was used to complete the proposed project:

- a. Analysis of previous manual system
- b. User review of proposed system
- c. Design of database
- d. Design of data entry utilities
- e. Validation of data entry utilities
- f. Redesign of data entry utilities
- g. Design of query interface
- h. Validation of query interface
- i. Redesign of query interface
- j. User Training
- k. Installation of final system

## Software Required

-----

dBASE PLUS III, Version 1.1 or higher.

## Hardware Required

-----

RCIS was designed for an IBM PC/XT, PC/AT or MS-DOS compatible configured with one floppy diskette drive and one hard disk drive.

## User Documentation

-----

System documentation consists of a User's Manual and a Technical Reference Manual. The User's Manual is intended to assist the users in operating and maintaining the system. The Technical Manual provides documentation for the design of the databases and the system software. The Technical Manual is intended primarily as a programmer's maintenance guide.

### 3. TASK PROGRESS

#### Analysis of Previous System -----

The initial interviews with members of the detachment staff were conducted by Ron Crane, Gary McAlum, Gary Talbot and myself during January and February 1987. The purpose of the interviews was to collect background information to design a mainframe database application for the MIS 531B class project.

We conducted the interviews in two phases. The first phase concentrated on the executive staff's view of how the database system could automate the manual compilation of cadet data used to complete reports and forms required by Headquarters AFROTC. The executive staff concluded that the proposed database system would drastically reduce the amount of time necessary to organize the required data and would give the entire staff more time to dedicate directly to the cadets. The executive staff was so excited about the project that they immediately put in a requisition for the Zenith PC micro-computer system which would be used to implement the PC based system.

The second analysis phase consisted of interviews with individual members of the executive and administrative staff. Data field requirements were obtained from each individual and, after a few data analysis and user review sessions, the required group of data fields was agreed upon. Each person on the staff submitted the types of data groupings (queries) they performed and we designed a set of query functions to meet the staff's requests. The staff reviewed the query functions and

minor modifications were made.

#### User Review of Proposed System

-----

The entire staff was briefed on the overall functional requirements developed from the information gathered and their final approval was obtained before we began the database system design. We advised the staff that the prototype system would be developed on the university mainframe computer system and that the final deliverable system would be transported to the office microcomputer using dBASE III PLUS.

In April 1987, the prototype system was demonstrated for the detachment staff. We reviewed the previously accepted system requirements and discussed possible changes required for the micro-based system. It was enthusiastically received by all members of the staff and approval was given to begin conversion onto the Zenith PC system.

#### Design of the Database

-----

The database design used for the mainframe application required revision before it could be efficiently implemented on a microcomputer. The prototype system structure was analyzed to determine the best structure which would provide optimum performance in the dBASE III PLUS environment on the PC. The system was required to support two classes of cadet records: cadets currently enrolled in the AFROTC program and cadets who either disenrolled or successfully completed the program. By separating the active and inactive records, system performance

could be substantially improved. In addition, this separation would also simplify query processing. For a detailed discussion of the database design, refer to the Technical Reference Manual.

#### Design of Data Entry Utilities

-----

The initial utilities incorporated into the system included the basic data entry and maintenance utilities, i.e. Add, Edit, View, Delete and Transfer. The administrative staff was concerned with the magnitude of data entry effort required to audit and enter 200 cadets (over 500 characters each, for a total of over 100,000 characters). To facilitate this effort, a data entry form was designed which matched the data field order on the system's data entry screens. This form would be used to gather cadet data from the various cadet files for entry into the system. In the future it would be used to enter a new cadet's data which could be gathered from an initial interview or a package of background information received from the cadet. A paging function was incorporated to allow the user to easily locate the data entry screen which contained the data fields they needed to update.

#### Validation of Data Entry Utilities

-----

The Zenith PC I developed the system on was located in the detachment's administrative office, so I had the opportunity for the staff to informally review the system's progress almost on a weekly basis. Several semi-formal review sessions were conducted to familiarize the staff with the evolving system



capabilities. During these sessions, additional database field requirements were identified for inclusion in the cadet database files.

#### Redesign of Data Entry Utilities

-----

The additional fields were added and corrections to the data entry utilities were completed in July 1987. By this time, the administrative staff had begun to gather data from the cadet files and had completed approximately 10 data entry forms.

#### Design of Query Interface

-----

In the past, the detachment staff had been unable perform numerous desirable ad hoc queries because the manpower required to manually search the existing file system was prohibitive. RCIS provides the database structure that should facilitate processing queries. Unfortunately, detachment staff personnel have no experience with the dBASE III PLUS command language. To handle the staff's future query processing requirements, a general-purpose friendly interface was essential.

The query requirements gathered during the prototype design were used as a basis for the design of the query input screens. The query input screens were designed to allow the user to constrain predefined data fields or set ranges of values for the predefined fields by using relational operators. The predefined fields on the input screen are designed to give the user maximum flexibility in processing queries for that particular type of query. The interface is restricted in the sense that it only

allows the user to specify AND conditions, but for almost all cases, this is not a severe restriction. In addition to allowing the user to process a wide-variety of query requirements, the query interface output screens and reports were meticulously designed to efficiently use the space provided on the screens and reports.

#### Validation of Query Interface

-----

As was the case with the data entry utilities, I had the opportunity for the staff to informally review the system's progress almost on a weekly basis. Once again, semi-formal review sessions were conducted to familiarize the staff with the evolving capabilities of the query interface. During these sessions, additional predefined input field requirements were identified for inclusion on the query input screens and data fields in the query output formats were identified for addition and deletion.

#### Redesign of Query Interface

-----

The predefined fields were added to the query input screens and corrections were completed in August 1987. By this time the administrative staff had completed approximately 30 cadet data entry forms.

#### User Training

-----

User training for the data entry utilities and the query interface was conducted during the first week of September 1987.

Most of the staff had little previous experience with micro-computers but they all expressed a willingness to learn.

We reviewed each of the basic data entry procedures and walked through a few example entry sessions to show the staff how to navigate their way through the system. We reviewed the basic query input and output formats by performing some example queries on test data previously entered onto the database. I demonstrated how each of the 10 high level queries were designed to provide flexibility in performing more specific queries in addition to their primary stated function.

#### Installation of Final System

-----

The final system is a Run-Time+ version of the data entry utilities and the query interface. The Run-Time+ utility encrypts and compresses the dBASE III PLUS source code and provides a faster running system. The actual source code will be stored on two separate floppy disks in a secure location and will not be available on the hard disk or the system load disks. This will ensure that no unauthorized changes can be made to the source code.

#### 4. FINAL REMARKS

##### Contribution to the Field of MIS

-----

This master's project has produced a custom designed database system that provides straightforward data entry utilities and a nonprocedural, user-friendly interface for query processing. The basic system menu was coded in assembly language by Stephen M. Curran and it duplicates the flexible ASSIST level menu system provided by dBASE III PLUS.

The project demonstrates the effectiveness of employing the following techniques in designing a system:

- a. Initial analysis performed using a formal requirements collection approach.
- b. Initial system design and confirmation of system requirements accomplished by using a prototype.
- c. Soliciting user validation of the system performance during critical phases in the system development.

##### Practical Experience Gained

-----

The project provided experience in designing a database application from two perspectives: first from the mainframe perspective and then for the micro-computer environment. I was surprised at the number of database structure changes and database language function changes required to convert the mainframe database design into a design which would provide optimum performance in the dBASE III PLUS micro-computer environment.

Although the detachment staff was very cooperative in this

endeavor, their lack of experience with micro-computers often led to misunderstandings as to what they really wanted from the system. However, this project was a success because those misunderstandings were overcome by allowing the individual staff members to participate and to shape the direction of the system.

USER'S MANUAL  
FOR  
ROTC CADET INFORMATION SYSTEM ( RCIS )  
VERSION 1.10

BY  
Carter L. Frank

A Report Submitted in Partial Fulfillment of the  
Requirements for the Degree of Master of Science  
(Management Information Systems)  
in The University of Arizona

1987

Master Committee:  
Dr. Sudha Ram

## TABLE OF CONTENTS

Title page.....	i
Table of Contents.....	ii
Table of Figures.....	iii
1.0 Introduction.....	1
1.1 Overview.....	1
1.2 Getting Started.....	2
2.0 The Menu Interface.....	5
2.1 Function Menu.....	5
2.2 Group Menu.....	6
2.3 Record Menu.....	7
2.4 Access Key Input.....	9
2.5 Query Selection Menu.....	12
2.6 Output Media Menu.....	13
3.0 Specifying the Record to Process.....	14
3.1 Adding, Editing, and Viewing Records.....	15
3.2 Deleting Records.....	19
3.3 Transferring Records.....	22
3.4 Database Queries.....	22
4.0 Maintaining the Database.....	27
4.1 Data Entry Techniques.....	27
4.2 Convention Establishment.....	28
4.3 Data Audits.....	29
4.4 Periodic Backups.....	30
4.5 Reloading the Database after System Disk Failure.....	31
5.0 Database Program and Support Files.....	32
APPENDIX (Query Input screens and Report formats)	

## TABLE OF FIGURES

1.1	RCIS Log-on screen.....	3
1.2	RCIS Function Menu.....	4
2.1	RCIS Group Menu.....	7
2.2	RCIS Record Menu.....	8
2.3	Transfer function menu sequence.....	8
2.4	Add function menu sequence.....	10
2.5	Edit, View & Delete functions menu sequence.....	11
2.6	Edit function access key change request.....	11
2.7	RCIS Query Selection Menu.....	12
2.8	RCIS Output Media Menu.....	13
3.1	Initial data entry/view screen for Master record.....	16
3.2	Data view screen for Pay records.....	16
3.3	Data entry screen for Pay records.....	18
3.4	Deleting a Master record (from Add function).....	20
3.5	Delete screen for Pay records.....	21
3.6	Sample Query input screen.....	25
3.7	Sample Query output (80-column format).....	26
3.8	Sample Query output (132-column format).....	26
4.1	Database utilities menu.....	31



## 1.0 INTRODUCTION.

This manual provides operating instructions for the ROTC Cadet Information System (RCIS), version 1.10. In the sections that follow, you'll be introduced to RCIS data files, and you'll be shown how to access data entry, query and maintenance functions. Additional technical data is available in Section 5.

### 1.1 OVERVIEW.

RCIS consists of two major groupings of files. Active files contain data on cadets currently enrolled in the AFROTC program. Inactive files contain data on cadets who either disenrolled or successfully completed the program. Within each of these file groupings there are two major subdivisions:

- a. Cadet Master file.
- b. Cadet Pay file.

The cadet master file contains personal, administrative, academic and corps information for each cadet. The cadet master file is the most important database file because all the other database files are used to support the master file information. The cadet master record can be thought of as the parent record for the cadet pay records, therefore, a master record must be created before any associated pay records can be added to the database. The cadet pay records contain required pay data for cadets who are contractually obligated to the AFROTC program. There can be multiple pay records for any one cadet (current

system limitation is 16 pay records but system could be modified to allow an unlimited number). The remaining database files are really tables of information created to facilitate an efficient database design. A description of each of these files is given as follows:

- a. Class Enrollment Totals - Contains an entry for each aerospace studies class with an associated total enrollment for that class.
- b. Weight Standards - Contains maximum and minimum allowable weight standards (male & female) associated with a given height.
- c. Aerobics Run Standards - Contains maximum allowable run times (male & female) associated with a given age category.
- d. WPSS Multipliers - Contains multiplier values used in calculating each cadet's WPSS score.

RCIS provides you with the functions required to enter, update (or edit), view, delete or transfer cadet master and pay records. The system also allows you to 'ask' questions about the information stored in the database. In the next section you'll be shown how to start RCIS and how to use basic system features.

## 1.2 GETTING STARTED.

To install the program, insert the RCIS system 1 diskette in drive A and type the following: `COPY A:\DBASE\*. * C:\DBASE\*. *` This command will copy basic program files to the dBASE III PLUS subdirectory. Next, insert the RCIS system 2 diskette in drive A and type the following: `COPY A:\DBASE\*. * C:\DBASE\*. *` This

will copy database definition files and other required files to the dBASE III PLUS subdirectory.

To start RCIS you must load dBASE III PLUS. Ensure that the computer system is in the dBASE III PLUS subdirectory by typing the following: `CD C:\DBASE` When the system prompt returns simply type `DBASE` and wait for dBASE III PLUS to be loaded. Once dBASE III PLUS has been loaded, press the <Esc> key. This will move the cursor from the ASSIST menu and place it at the bottom left hand corner of the text window. To start RCIS, type `DO RCIS`. After a short delay you should see the initial RCIS screen shown below.

ROTC CADET INFORMATION SYSTEM (RCIS)
Version 1.10 by Carter L. Frank The University of Arizona Department of Management Information Systems Copyright (C) 1987
INITIALIZING RCIS

Figure 1.1 RCIS Log-on screen.

While the log-on screen is displayed, the program starts to INITIALIZE information required to operate the system. This set-up process will require about 15 seconds to complete. Once INITIALIZATION is finished, the log-on screen will be replaced by the screen shown below.

ROTC CADET INFORMATION SYSTEM (RCIS)	
FUNCTION	
Add	
Edit	
View	
Delete	
Transfer	
Query	
dBASE	
Exit	

SELECT FUNCTION
-----------------

Figure 1.2 RCIS Function menu.

You've started RCIS and are now ready to begin data entry. The next sections will discuss how to access particular functions to enter or manipulate RCIS records.

## 2.0 THE MENU INTERFACE.

RCIS allows you to specify the type of processing you want to do by selecting from a menu. The menu interface was designed to be similar to the existing dBASE III PLUS ASSISTANT interface. This section discussed how to make selections using the menu interface.

### 2.1 FUNCTION MENU.

The function menu is the first menu presented to you after INITIALIZATION has been completed (see Figure 1.2). You will use this menu to designate the type activity you wish to perform. You are presented with 8 options:

- a. Add - Choose this function if you wish to create a new record.
- b. Edit - Select this option if you wish to update or make changes to a specific record that already exists.
- c. View - Choose this option if you desire to look at a specific record, but don't want to alter any information. This function is used to prevent inadvertent data alterations that might occur if you had selected edit.
- d. Delete - Select this function to delete a specific record. If a cadet master record is selected for deletion, then all associated cadet pay records for that master record are also deleted.
- e. Transfer - Choose this option to move a cadet master record and all its associated pay records either from the active to the inactive file, or from the inactive to the active file.
- f. Query - Select this option to perform queries on the database files.

- g. dBASE - Select this option to exit RCIS and return to dBASE III PLUS.
- h. Exit - Select this option to exit RCIS and return to the computer system prompt.

To select a function from the menu, press either the up arrow key or the down arrow key (located on the key pad). Continue pressing the up or down arrow key until the function you want to select is highlighted. You complete your function selection by pressing the <Enter> key. If you inadvertently made an erroneous choice, you can return to the function menu by later pressing the <Esc> key.

#### NOTE

If the highlight doesn't change, check the NUM LOCK light. If it is illuminated, you're in number keypad mode. Press the NUM LOCK key to activate the cursor keypads.

## 2.2 GROUP MENU.

After you've selected a function, another menu will appear next to the function menu (see Figure 2.1). This menu is used to select records from either the active or inactive database files. Again, press the cursor keys to highlight your choice and press the <Enter> key. If you have made a mistake in choosing either a function or a group, you can "roll-back" to a previous menu by pressing the <Esc> key until the desired menu becomes active.

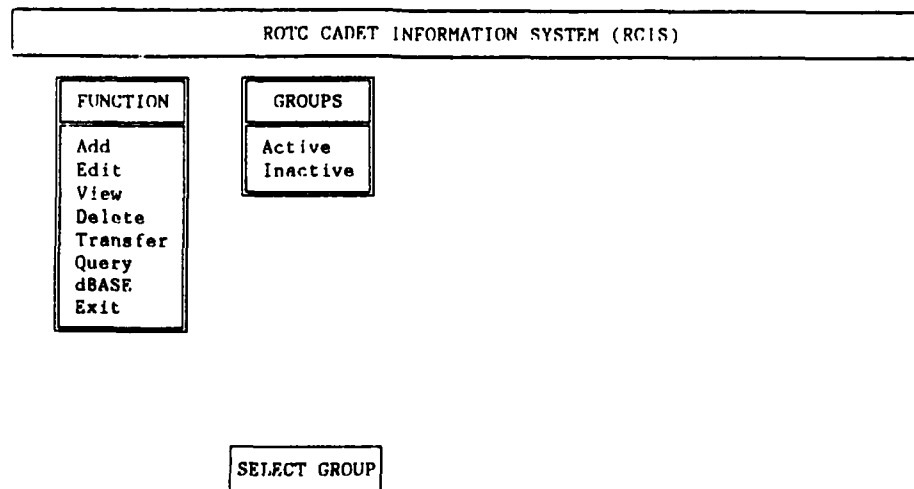


Figure 2.1 RCIS Group menu

### 2.3 RECORD MENU.

After selecting a database group, another menu will appear on the screen. This new menu is used to select the record type that you want to access. As shown in Figure 2.2, there are two record types to choose from (Cadet Master and Cadet Pay). Again, you select the desired record type by highlighting your choice using the cursor keys and pressing the <Enter> key. If your previous menu selections are incorrect, press the <Esc> key to "roll back" to the menu that must be corrected.

There are only two function selections that will generate a different sequence of menus than shown in Figure 2.2. If your function choice was Transfer, an access key input request will appear in the bottom lefthand corner of your screen as shown in Figure 2.3. The menu shown in Figure 2.7 will appear if you selected the Query function.

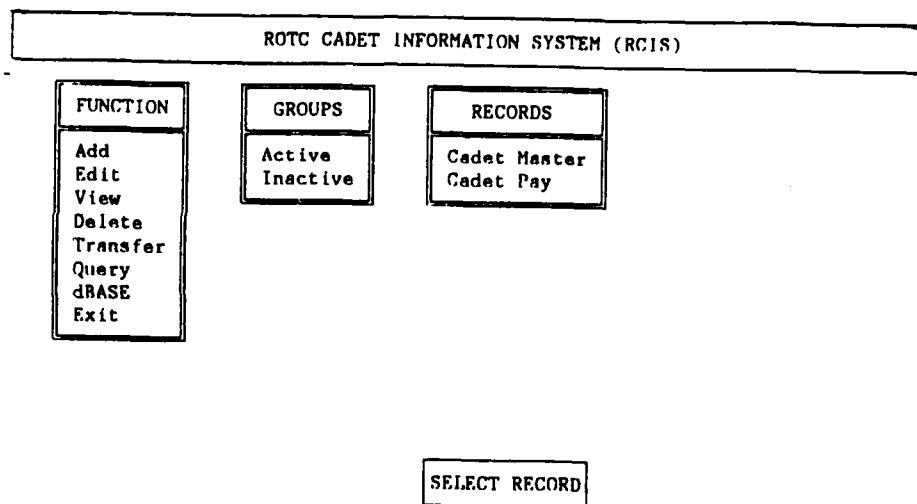


Figure 2.2 RCIS Record menu

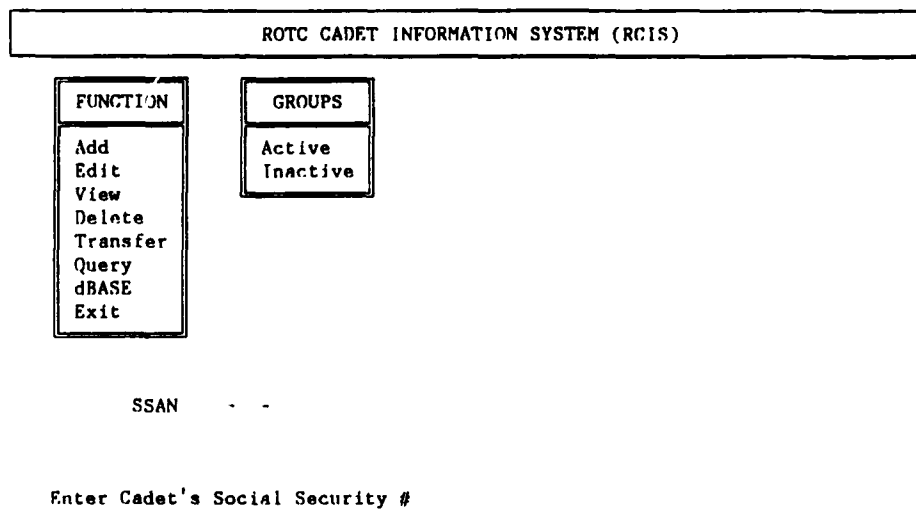


Figure 2.3 Transfer function menu sequence



## 2.4 ACCESS KEY INPUT.

After you have selected a record type (for Transfer function after you have selected a group type), an access key input request will appear in the bottom lefthand corner of your screen. For the Add and Transfer functions the request will appear as shown previously in Figure 2.3 and in Figure 2.4. For the Edit, View and Delete functions the request will appear as shown in Figure 2.5. The data items you will be entering (social security number, first name, middle name or last name) are known as access keys. Basically you can consider a database to be an extended file cabinet that is very thoroughly cross-referenced.

For example, you might like to locate a cadet record in your manual file system, but all you have is the social security number. If the file system is arranged alphabetically by cadet name, you might not be able to find the folder; however, if you had a card file that cross-references social security numbers with names, you could easily locate the required record. Databases use this same approach. Special files (called index files) are used to cross-reference the location of a particular record. These indices allow you to use various data items as keys to finding the desired record.

So, before we can locate a record in our database we must specify how to look for it. The access key input request allows you to locate records in two different ways (except for the Add and Transfer functions). You will be able to locate records by using the cadet's social security number or by using a portion of

their name that uniquely identifies the cadet from all the others on the database. If you enter a social security number and a name, the system will default to use only the social security number.

After the access key input request appears on the screen, you can not "roll back" to a previous menu; however, you can still abort the operation by pressing the <Esc> key before entering any data in the highlighted fields.

#### NOTE

If you have selected the Edit function and the system has successfully located the record you want to edit, the system will ask you if you would like to change the cadet's social security number (perhaps it was initially entered incorrectly). If you respond by entering a <Y> then an access key change request will appear as shown in Figure 2.6. You will be shown the current access keys (social security number and name) for the record and be given the opportunity to change only the social security number.

ROTC CADET INFORMATION SYSTEM (RCIS)		
<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"><b>FUNCTION</b></div> <div style="border: 1px solid black; padding: 2px;"><div>Add</div><div>Edit</div><div>View</div><div>Delete</div><div>Transfer</div><div>Query</div><div>dBASE</div><div>Exit</div></div>	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"><b>GROUPS</b></div> <div style="border: 1px solid black; padding: 2px;"><div>Active</div><div>Inactive</div></div>	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px;"><b>RECORDS</b></div> <div style="border: 1px solid black; padding: 2px;"><div>Cadet Master</div><div>Cadet Pay</div></div>
SSAN    -    -		
Enter Cadet's Social Security #		

Figure 2.4 Add function menu sequence

ROTC CADET INFORMATION SYSTEM (RCIS)		
<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">FUNCTION</div> <div style="border: 1px solid black; padding: 2px;"> Add Edit View Delete Transfer Query dBASE Exit </div>	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">GROUPS</div> <div style="border: 1px solid black; padding: 2px;"> Active Inactive </div>	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">RECORDS</div> <div style="border: 1px solid black; padding: 2px;"> Cadet Master Cadet Pay </div>

SSAN        - - -  
First Name  
Middle Name  
Last Name  
Enter Cadet's Social Security #    OR    Name.

Figure 2.5    Edit, View & Delete functions menu sequence

ROTC CADET INFORMATION SYSTEM (RCIS)		
<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">FUNCTION</div> <div style="border: 1px solid black; padding: 2px;"> Add Edit View Delete Transfer Query dBASE Exit </div>	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">GROUPS</div> <div style="border: 1px solid black; padding: 2px;"> Active Inactive </div>	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">RECORDS</div> <div style="border: 1px solid black; padding: 2px;"> Cadet Master Cadet Pay </div>

SSAN 111-11-1111

First Name CARTER

Middle Name LEROY

Last Name FRANK

New SSAN        - - -

Enter New SSAN    or

Press FSC to Continue.

Figure 2.6    Edit function access key change request

## 2.5 QUERY SELECTION MENU.

The query selection menu will appear as shown in Figure 2.7 if you have selected the Query function. This menu allows you to select the particular query type you need to process your database questions. The method for selecting from this menu is the same as the previous menus, i.e. choose selection using the cursor keys and then press the <Enter> key and if previous menu selections are incorrect, press the <Esc> key to "roll back" to the menu that must be corrected. Each type of query has its own query input form (see Appendix A) which shows you the constraint fields for that particular type of query. These forms are discussed in more detail in Section 3.4 DATABASE QUERIES.

ROTC CADET INFORMATION SYSTEM (RCIS)																									
<table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th style="text-align: left; padding: 2px;">FUNCTION</th></tr></thead><tbody><tr><td>Add</td></tr><tr><td>Edit</td></tr><tr><td>View</td></tr><tr><td>Delete</td></tr><tr><td>Transfer</td></tr><tr><td>Query</td></tr><tr><td>dBASE</td></tr><tr><td>Exit</td></tr></tbody></table>	FUNCTION	Add	Edit	View	Delete	Transfer	Query	dBASE	Exit	<table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th style="text-align: left; padding: 2px;">GROUPS</th></tr></thead><tbody><tr><td>Active</td></tr><tr><td>Inactive</td></tr></tbody></table>	GROUPS	Active	Inactive	<table border="1" style="width: 100%; border-collapse: collapse;"><thead><tr><th style="text-align: left; padding: 2px;">QUERY TYPE</th></tr></thead><tbody><tr><td>WPSS Info</td></tr><tr><td>Schlrshp Qual</td></tr><tr><td>DOC Fiscal Yr</td></tr><tr><td>AS Class Info</td></tr><tr><td>2-Yr Pgm Cand</td></tr><tr><td>Com Date Susp</td></tr><tr><td>Schlrshp Expr</td></tr><tr><td>Weigh/Aerobic</td></tr><tr><td>Individual</td></tr><tr><td>Pay Info</td></tr></tbody></table>	QUERY TYPE	WPSS Info	Schlrshp Qual	DOC Fiscal Yr	AS Class Info	2-Yr Pgm Cand	Com Date Susp	Schlrshp Expr	Weigh/Aerobic	Individual	Pay Info
FUNCTION																									
Add																									
Edit																									
View																									
Delete																									
Transfer																									
Query																									
dBASE																									
Exit																									
GROUPS																									
Active																									
Inactive																									
QUERY TYPE																									
WPSS Info																									
Schlrshp Qual																									
DOC Fiscal Yr																									
AS Class Info																									
2-Yr Pgm Cand																									
Com Date Susp																									
Schlrshp Expr																									
Weigh/Aerobic																									
Individual																									
Pay Info																									
<table border="1" style="margin: auto; padding: 5px;"><tr><td>SELECT QUERY</td></tr></table>			SELECT QUERY																						
SELECT QUERY																									

Figure 2.7 RCIS Query Selection menu

## 2.6 OUTPUT MEDIA MENU.

The output media menu appears after the Query selection menu as shown in Figure 2.8. This menu allows you to specify the device and format to be used to display the results of query processing. You can select from one of three options:

- a. 80-column monitor - This option will direct all output to the screen.
- b. 80-column printer - This option will direct all output to the printer using standard font (12 pitch) and standard paper size.
- c. 132-column printer - This option will direct all output to the printer using compressed mode (17 pitch) and standard size paper.

The method for selecting from this menu is the same as the previous menus, i.e. choose selection using the cursor keys and then press the <Enter> key and if previous menu selections are incorrect, press the <Esc> key to "roll back" to the menu that must be corrected.

ROTC CADET INFORMATION SYSTEM (RCIS)			
<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">FUNCTION</div> Add Edit View Delete Transfer Query dBASE Exit	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">GROUPS</div> Active Inactive	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">QUERY TYPE</div> WPSS Info Schlrshp Qual DOC Fiscal Yr AS Class Info 2-Yr Pgm Cand Com Date Susp Schlrshp Expr Weigh/Aerobic Individual Pay Info	<div style="border: 1px solid black; padding: 2px; margin-bottom: 5px; text-align: center;">QUERY OUTPUT</div> 80-Col Screen 80-Col Printer 132-Col Printer
<div style="border: 1px solid black; padding: 2px; display: inline-block;">SELECT OUTPUT MEDIA</div>			

Figure 2.8 RCIS Output Media menu

### 3.0 SPECIFYING THE RECORD TO PROCESS.

Once you have selected a function, group, record type and/or entered an access key, the system prints a message informing you that it is opening requested files and that it is searching for the designated record. If you've selected the Add function, the system will check to ensure that no duplicate record already exists because you are only permitted to create a record with a unique access key. If you've selected the Transfer function, the system will check to ensure that no duplicate record already exists on the destination file because you are not permitted to transfer a record if it would cause duplicates to exist on the destination file. For the Edit, View and Delete functions, you're searching to find a record that should already exist.

The system will inform you of the result of the search if a special case has been encountered. For example, if you've selected the Add function and a record has already been assigned to the access key you've input, the system bell will sound and an appropriate message will be displayed. In another situation, you may have selected the Edit function and the system is searching for the designated record, but was unable to locate it (usually because of a typographical error). The system bell will sound and a MASTER (or PAY) RECORD NOT FOUND message will be displayed. Another special case can occur when you've specified a non-unique access key, e.g. LAST NAME = SMITH. In this instance the system will advise you if more than one record exists.

If a system message is displayed, you are given further

instructions. For example, you might be asked if you wish to try again or you may simply be asked to press any key to continue. Once you've ended a transaction, the menu screen will reappear and you'll be asked if you want to continue in the same mode. If you answer <Y>, then you'll be prompted to input a new access key value. If you answer <N>, then the system will close its working files and you'll be returned to the select function menu. At this point, you can choose another function and continue processing or you can elect to exit RCIS.

### 3.1 ADDING, EDITING AND VIEWING RECORDS.

If the search operation has been successfully concluded, the next screen that appears will be the initial data entry or data view screen (see Figure 3.1 for master record and Figure 3.2 for pay record). You are then free to enter data or modify data in any field that is highlighted (no highlighted data fields on the View function screens). Use the cursor keys to maneuver around the screen (cursor can only be moved to highlighted fields).

INDIVIDUAL CADET DATA - PERSONAL INFORMATION				(Page 1 of 4)
SSAN 222-22-2222		Matric #		
First Name			Age	Sex
Middle Name			Birthdate / /	
Last Name				
<b>LOCAL</b>				
Street Address				
City			Phone -	
Zip Code -				
<b>PERMANENT</b>				
Street Address				
City			Phone ( ) -	
State Zip Code -				

Figure 3.1 Initial data entry/view screen for Master record

FRANK, C L. INDIVIDUAL CADET DATA - PAY INFORMATION									
REC #	BEGINNING PAY DATE	ENDING PAY DATE	TUITION	RESID (I OR O)	BOOK FEES	FT DAYS	ATP DAYS	FSP DAYS	
1	01/09/85	31/12/85	1300.00	0	100.00	0	0	0	
2	01/01/86	31/05/86	600.00	I	150.00	0	0	0	
3	01/09/86	31/12/86	700.00	I	200.00	0	0	0	
4	01/01/87	31/05/87	800.00	I	250.00	0	0	0	
5	01/06/87	31/08/87	0.00		0.00	28	14	14	
6	01/09/87	31/12/87	900.00	I	300.00	0	0	0	
7	01/01/88	31/05/88	1000.00	I	350.00	0	0	0	
8	01/09/88	31/12/88	1100.00	I	425.00	0	0	0	
9	01/01/89	31/05/89	1200.00	I	450.00	0	0	0	
10	01/06/89	31/08/89	750.00	I	175.00	0	0	0	

PRESS ANY KEY TO RETURN TO MAIN SELECTION SCREEN

Figure 3.2 Data view screen for Pay records



### 3.1.1 MASTER RECORDS.

The master record data forms are four pages (screens) long and you can advance to the next page by pressing the <PgDn> key or you can go back to the previous page by pressing the <PgUp> key. If you <PgDn> past the last page or <PgUp> past the first page, the record transaction will be terminated. Another way to terminate a record transaction is to press the <Ctrl> <End> keys. During editing, you can abort any changes and restore the record to its initial state by pressing the <Esc> key.

### 3.1.2 PAY RECORDS.

All pay records associated with the input access key will be displayed on the same screen (see Figure 3.3). If you've selected the Add function, you can add the pay record input data to the database by pressing one of the following key sequences: <PgUp>, <PgDn>, <Esc>, <Ctrl><End>. If you've selected the Edit function, the system will prompt you to enter the corresponding record number for the pay record you would like to change (record numbers are listed on the screen). After you've entered the desired record number and pressed the <Enter> key, the system will highlight the pay record you have selected. The new pay record input data can be added to the database by pressing one of the following key sequences: <PgUp>, <PgDn>, <Esc>, <Ctrl><End>. The system will unhighlight the pay record and prompt you for another selection.

## NOTES

A <Y> is required in the ADD field for the pay record to be added to the database. A <N> in the ADD field will cancel the add and it is the only way to terminate this function.

The beginning and ending dates for each pay record are used to define the pay period for that record. There is extensive error checking done to ensure that these pay periods do not overlap. In other words, the system will not allow you to input pay dates which would cause pay periods to overlap.

FRANK, C L                      INDIVIDUAL CADET DATA - PAY INFORMATION									
ADD	REC #	BEGINNING PAY DATE	ENDING PAY DATE	TUITION	RESID (I OR O)	BOOK FEES	FT DAYS	ATP DAYS	FSP DAYS
	1	01/09/85	31/12/85	1300.00	O	100.00	0	0	0
	2	01/01/86	31/05/86	600.00	I	150.00	0	0	0
	3	01/09/86	31/12/86	700.00	I	200.00	0	0	0
	4	01/01/87	31/05/87	800.00	I	250.00	0	0	0
	5	01/06/87	31/08/87	0.00		0.00	28	14	14
	6	01/09/87	31/12/87	900.00	I	300.00	0	0	0
	7	01/01/88	31/05/88	1000.00	I	350.00	0	0	0
	8	01/09/88	31/12/88	1100.00	I	425.00	0	0	0
	9	01/01/89	31/05/89	1200.00	I	450.00	0	0	0
	10	01/06/89	31/08/89	750.00	I	175.00	0	0	0
Y	11	01/01/01	01/01/01	0.00		0.00	0	0	0

ENTER 'Y' IN ADD FIELD TO ADD PAY RECORD. ENTER 'N' IN ADD FIELD TO CANCEL ADD.

Figure 3.3 Data entry screen for Pay records

### 3.2 DELETING RECORDS.

The delete function has been provided to allow you to remove a record from the database. During data audits, you should look for extraneous or unwanted records. These unwanted records should be deleted from the system because they will eventually cause the system to become less efficient. Their presence will require longer search times to locate valid records for processing.

#### 3.2.1 MASTER RECORDS.

There are two ways you can delete a master record. If you have made a lot of mistakes in entering data during record creation (Add function) or have just decided not to add it, you can delete the record before it is added to the system by pressing the <Ctrl> <U> keys. This marks the record for deletion. The system will indicate that the record was marked for deletion by placing the symbol DEL in the status line (see area labeled 1 in Figure 3.4). After you exit the data entry form, RCIS will ask you if you want to delete the record. Enter <Y> if you want to delete or enter <N> if you want to retain the record.

Once a master record has been added, the only way to remove it is by using the Delete function. To delete a master record, select the Delete function, specify the group (inactive or active) and specify Cadet Master record type. The system will prompt you to enter the access key value for the record. After

conducting a record search, the system will display the record for confirmation. You can scroll through the record pages by using the <PgUp> and <PgDn> keys. When you are finished viewing the record press the <Ctrl> <End> keys or page past either end of the record pages. The system will ask if you want to delete the record. Enter <Y> to delete the record or press <N> to retain. After deleting the master record, the system will delete all pay records associated with that master record.

1 Del Caps

INDIVIDUAL CADET DATA - PERSONAL INFORMATION				(Page 1 of 4)						
SSAN 333-33-3333		Matric #								
First Name	Age	Sex								
Middle Name	Birthdate / /									
Last Name										
<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;">LOCAL</div>										
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%; border-bottom: 1px solid black; padding-bottom: 5px;">Street Address</td> <td style="width: 40%; border-bottom: 1px solid black; padding-bottom: 5px;">Phone -</td> </tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;">City</td> <td></td> </tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;">Zip Code -</td> <td></td> </tr> </table>					Street Address	Phone -	City		Zip Code -	
Street Address	Phone -									
City										
Zip Code -										
<div style="border: 1px solid black; display: inline-block; padding: 2px 10px;">PERMANENT</div>										
<table style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 60%; border-bottom: 1px solid black; padding-bottom: 5px;">Street Address</td> <td style="width: 40%; border-bottom: 1px solid black; padding-bottom: 5px;">Phone ( ) -</td> </tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;">City</td> <td></td> </tr> <tr> <td style="border-bottom: 1px solid black; padding-bottom: 5px;">State Zip Code -</td> <td></td> </tr> </table>					Street Address	Phone ( ) -	City		State Zip Code -	
Street Address	Phone ( ) -									
City										
State Zip Code -										

Figure 3.4 Deleting a Master record (from Add function)

### 3.2.2 PAY RECORDS.

All pay records associated with the input access key will be displayed on the same screen (see Figure 3.5). You will be prompted to enter a <Y> in the DEL field of each pay record you want to delete. When you have finished "marking" the desired pay records for deletion, press one of the following key sequences to start the deletion: <PgUp>, <PgDn>, <Esc>, or <Ctrl><End>. The system bell will sound and a ONLY DELETING "MARKED" RECORDS message will be displayed until deletion is complete.

FRANK, C L		INDIVIDUAL CADET DATA - PAY INFORMATION							
DEL.	REC #	BEGINNING PAY DATE	ENDING PAY DATE	TUITION	RESID (I OR O)	BOOK FEES	FT DAYS	ATP DAYS	FSP DAYS
N	1	01/09/85	31/12/85	1300.00	O	100.00	0	0	0
N	2	01/01/86	31/05/86	600.00	I	150.00	0	0	0
N	3	01/09/86	31/12/86	700.00	I	200.00	0	0	0
N	4	01/01/87	31/05/87	800.00	I	250.00	0	0	0
N	5	01/06/87	31/08/87	0.00		0.00	28	14	14
N	6	01/09	31/12/87	900.00	I	300.00	0	0	0
N	7	01/01/88	31/05/88	1000.00	I	350.00	0	0	0
N	8	01/09/88	31/12/88	1100.00	I	425.00	0	0	0
N	9	01/01/89	31/05/89	1200.00	I	450.00	0	0	0
N	10	01/06/89	31/08/89	750.00	I	175.00	0	0	0

ENTER A 'Y' IN THE DEL FIELD FOR EACH PAY RECORD YOU WANT DELETED.

Figure 3.5 Delete screen for Pay records

### 3.3 TRANSFERRING RECORDS.

Overall system performance can also be improved if records for disenrolled or graduated cadets are transferred to the inactive files. The system provides the capability to transfer a master record and all associated pay records. The process is very similar to deleting a master record. First, select the Transfer function and indicate the current location of the record to be transferred (active or inactive file). After entering the record access key value, the system will search for and display the record. You can scroll through the record pages by using the <PgUp> and <PgDn> keys. When you are finished viewing the record press the <Ctrl> <End> keys or page past either end of the record pages. You will also be given the option of viewing the associated pay records. The system will then ask you if you want to transfer the record. Enter <Y> to transfer or enter <N> to cancel. If you opt to transfer the record(s), the system displays advisories as it accomplishes the requested processing.

### 3.4 DATABASE QUERIES.

The query interface is the work horse of RCIS. It allows you to ask questions of the database without having to learn the dBASE III PLUS command language. The query input screens collect your query requirements using a simple form that allows you to set search restrictions or constraints. This means you can specify a range of values for a field to be used in the database search.

To access the query interface, select the Query function, specify the file group (active or inactive), choose a query type and select an output media. The system will then present a query input form that allows you to specify the constraints required to satisfy your question. There are six basic symbols used to specify search requirements:

- a.    =    - Indicates you want to specify an "equal to condition" for the search. Using this symbol means "show me only those records with values equal to this condition."
- b.    <>   - Indicates you want to specify a "not equal to condition" for the search. Using this symbol means "show me only those records with values not equal to this condition."
- c.    >    - Indicates you want to specify a "greater than condition" for the search. Using this symbol means "show me only those records with values greater than this condition."
- d.    <    - Indicates you want to specify a "less than condition" for the search. Using this symbol means "show me only those records with values less than this condition."
- e.    >=   - Indicates you want to specify a "greater than or equal to condition" for the search. Using this symbol means "show me only those records with values greater than or equal to this condition."
- f.    <=   - Indicates you want to specify a "less than or equal to condition" for the search. Using this symbol means "show me only those records with values less than or equal to this condition."

To specify a query, simply enter the appropriate symbols in the highlighted operator fields and enter the desired values in the highlighted data fields. When you are finished, press the <PgDn> key. The system will ask two questions before it processes the query. First, the system will ask you if you want

to cancel your query. Enter <N> to continue or enter <Y> to cancel the query and return to the select function menu. If you choose to continue, the system will ask you if you want to make any corrections. Enter <N> to process the query or enter <Y> to return to the input form and make corrections. If you elect to submit the query, the system will then check to ensure that valid symbols were used to specify the question. If an error in symbol use is detected, you will be asked to modify the query input form. If no errors are detected, the system will process your query and display the results on the media selected for output.

Example: The Professor of Aerospace Studies wants a detailed listing of WPSS scores (greater than or equal to 75) and related information for all sophomore cadets enrolled in the AFROTC program.

Step 1. Select the Query function, the Active group, the WPSS Info query type and the Query Output of your choice.

Step 2. The WPSS Query input screen will appear and you can proceed to enter the required constraints for this query. Since we are only interested in sophomore cadets, we will have to constrain the AS CLASS field. In addition, we are only interested in the sophomores who have WPSS scores that are greater than or equal to 75, so, we will also have to constrain the WPSS Score field. Finally, the query requires a detailed listing so we need to enter a <2> in the Print Options field

Step 3. Enter the constraints and options so that the query input screen looks like the one in Figure 3.6. Press the <PgDn> key when you are finished and the system will give you the opportunity to cancel the query or to make changes to your input. If you respond with a <N> for both questions, the system will attempt to process your query. If there are database records which meet your constraints, your query output will look like Figure 3.7 (80-column format) or Figure 3.8 (132-column format).



## NOTE

Each of the operator field/data field constraint pairs entered on the screen will be used to form a search condition for that particular query. The system will locate only those records which satisfy all the constraints in the combined search condition, i.e. constraint 1 AND constraint 2 AND constraint 3 AND etc.

Finally, you can obtain a printed copy of screen output without selecting the printer option directly. Simply press the <Shift> <PrtSc> keys to direct screen output to the printer. Please note that you are limited to 80-column capacity when using the screen for output. The 132-column printer option will provide you with additional information associated with the particular type of query you are performing.

WEIGHTED POC SELECTION SYSTEM (WPSS) QUERY			
AS Class = 2			
WPSS Score >= 75			
Last Name			
SSAN - -			
Print Options			
Brief - 1 , Detailed - 2 2			
EXAMPLE	Query Item	Operators[<,>,<=>,<=>,<=>]	Query Values
	Last Name	>= * Absence of Operator < field defaults to '='	ANDERSON SMITH

Figure 3.6 Sample Query input screen

# WEIGHTED POC SELECTION SYSTEM(WPSS) REPORT

First Name CARTER	Last Name FRANK	WPSS Score 103.22	DC Rating 7	GPA Cum 3.50	SAT Cum 1200	AFOQT AcAp 80	AFOQT Quan 80	AFOQT Verb 80
		AS Class 3	AS Class Rank 10/ 1	GPA Sem 3.60	SAT Math 600	SAT Verb 600	Schlr Type 3.0	Pilot Licns Y
		DOB 05/10/58	Age 28	Phys Date 01/03/89	Grad Date 01/10/86	Comm Date 01/10/86		

Figure 3.7 Sample Query output (80-column format)

# WEIGHTED POC SELECTION SYSTEM(WPSS) REPORT

First Name CARTER	Last Name FRANK	WPSS Score 103.22	DC Rating 7	GPA Cum 3.50	SAT Cum 1200	AFOQT AcAp 80	AFOQT Quan 80	AFOQT Verb 80	AFOQT Pilot 80	AFOQT Nav 80	Cat Type 2	FY Rating 45	FSP Major MIS	FSP Date / /
		AS Class 3	AS Class Rank 9/ 23	GPA Sem 3.60	SAT Math 600	SAT Verb 600	Schlr Type 3.0	Pilot Licns Y	4-Yr Cadet N	Prior Serv N	Waiv Req N	Race C		
		DOB 05/10/58	Age 28	Phys Date 01/03/89	Grad Date 01/10/86	Comm Date 01/10/86	Form 48	Corps Auxiliaries						

Figure 3.8 Sample Query output (132-column format)

#### 4.0 MAINTAINING THE DATABASE.

This section discusses techniques and procedures that should be enforced to ensure the integrity of the database. These maintenance procedures include:

- a. Data entry techniques.
- b. Convention establishment.
- c. Data audits.
- d. Periodic backups.
- e. Reloading the database after system disk failure.

#### 4.1 DATA ENTRY TECHNIQUES.

The first step in ensuring the integrity of the information stored within the system is to enter it correctly initially. This is an important factor in the reliability of the database, because the computer has no idea that a cadet's social security number, for example, has been entered incorrectly. Later, when you ask the system to retrieve information using the cadet's correct social security number, it will not be able to find it. If enough data entry errors have been introduced to the system, the value of the database is compromised. Eventually, everyone will lose confidence in the system's ability to provide accurate information for their use.

While data entry is a very demanding task, it can also be a very tedious process. There are two recommendations that can help ensure that the number of entry errors are reduced or caught

before moving on to the next record. First, critically review what has been entered before you commit it to the system. This simple process can help you catch typographical errors that might otherwise be entered into the system. Second, take frequent breaks. Fatigue will cause you to lose concentration. Couple this with the repetitive nature of data entry and you have a situation that invites entry errors.

#### 4.2 CONVENTION ESTABLISHMENT.

The second step in ensuring data integrity is to establish conventions for data entry and enforce them. A convention is simply a standardized way of entering information. For example, you might decide that the cadet's academic major (a four-character field) should be entered using standardized codes. If the same academic major is entered using different coding, the system's integrity is reduced. Essentially, the entry must be explicitly the same because computers cannot identify things in context the way that a human does. For example, the computer cannot recognize that "EENG", "EEGR", "ELEN" and "ELCE" all refer to the same academic major (Electrical Engineering).

One approach that can be taken is to create a convention book that lists the "rules" for entering data into the system. You should address the use of punctuation, abbreviations, codes and any other areas of ambiguity that can arise. Once you establish conventions, you should enforce them.

#### 4.3 DATA AUDITS.

The third method to ensure data integrity is to accomplish a periodic data audit. This essentially means that you should obtain a listing of information in the system and examine it for typographical errors and convention violations. While there is no "hard and fast" rule governing the frequency of audits for a system, there are several general criteria that can be used. First, more frequent data audits should be performed if the data entry operator is inexperienced. Second, if the system is frequently updated or new records are added frequently, then data audits should be more frequent. If the data entry operator is experienced or if the database is fairly stable, then the frequency of audits can be minimized.

You can use the Query function to obtain listings to assist you during data audits. The advantage of using the Query function is that you can limit the number of records and fields being reviewed. For example, you can elect to audit academic data for freshman and sophomore cadets (AS\_CLASS = 1 or 2) by using the SCHOLARSHIP/ACADEMIC PERFORMANCE query to limit your data output. The most important factor is that the auditor examine the data critically. If errors are detected, use the Edit function to make the required corrections.

#### 4.4 PERIODIC BACKUPS.

Once you've expended the time and energy to enter and verify the data, you should take positive action to protect it from loss. You can do this by obtaining a backup of the entire contents of the system database files. RCIS includes a special program, RCISUTIL, that makes it very easy to obtain a full backup of essential files.

To invoke the backup utility, type DO RCISUTIL from within dBASE III PLUS. You will be presented with a menu that allows you to select either Back-up or Reload (see Figure 4.1). Select Back-up by pressing the cursor keys until the Back-up option is highlighted. Then press the <Enter> key. The system will tell you to insert a blank formatted diskette in drive A. After inserting the diskette, press any key. The program will automatically copy all required files to the backup diskette. If additional diskettes are required to obtain a full backup, the system will instruct you to insert other blank, formatted diskettes. It will continue processing until all required files have been copied.

After the backup is complete, label the diskette and enter the date of the backup. Then store the diskette in a safe place. It may be a good idea to make another backup of the system and store it in a remote location. This can prove helpful if the first backup copy is lost or destroyed.

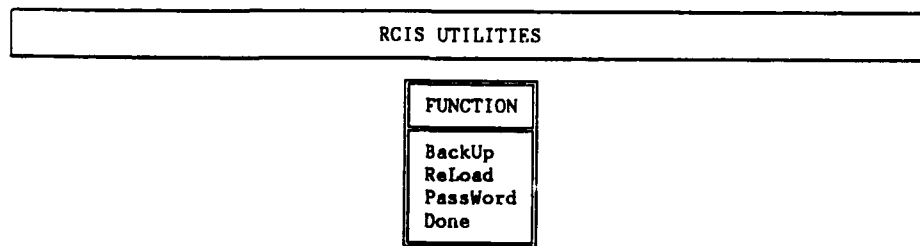


Figure 4.1 Database utilities menu using RCISUTIL

#### 4.5 RELOADING THE DATABASE AFTER SYSTEM DISK FAILURE.

If there is a catastrophic failure of the system hard disk, you can recover the database by reloading the system from your backup. Once the system disk is replaced, reinstall dBASE III PLUS and the RCIS program files. Then execute the RCISUTIL program. Choose the Reload option. The system will advise you that this option will overwrite the current database. You can abort the process if you have inadvertently selected Reload. Otherwise, continue with the program. If you elect to continue, the system will ask you to enter the system password. If you enter the wrong password, the program returns to the selection menu. If you enter the correct password, the system prompts you to insert the most current backup diskette in drive A. After accomplishing this, press any key and the system will automatically copy all database files to the hard disk. If two or more diskettes were required for the backup, the system will prompt you to insert the additional diskettes.

## 5.0 DATABASE PROGRAM AND SUPPORT FILES.

RCIS consists of the following program files:

- RCIS.PRG - This is the main controlling RCIS program file.
- RCIS\_P1.PRG - This file contains the RCIS initialization routines.
- RCIS\_P2.PRG - This file contains the following RCIS function routines: Add, Edit, View, Delete and Transfer.
- RCIS\_P3.PRG - This file contains all RCIS Query function routines.
- RCISUTIL.PRG - This is the main controlling program for the Backup and Reload utilities.
- RCISUTL2.PRG - This file contains the Backup and Reload function routines.

RCIS is supported by the following format files used to create the data entry and view format screens:

CDT\_M.FMT

CDT\_M\_VU.FMT

RCIS accesses the following database and index files (where X\_ symbolizes either A\_ for active file or I\_ for inactive file and T\_ is for table files):

Database File Name	Index File Name
X_CDT_MS.DBF	X_CGDT.NDX
	X_CLAS.NDX
	X_DCFY.NDX
	X_SCHA.NDX
	X_SEDT.NDX
	X_SSAN.NDX
	X_WPSS.NDX



X\_CDT\_PY.DBF

X\_PAY.NDX

X\_CDT\_CT.DBF

X\_ASCL.NDX

T\_CDT\_HW.DBF

T\_HGHT.NDX

T\_CDT\_RT.DBF

T\_AGEC.NDX

T\_CDT\_WP.DBF

-----

APPENDIX

### Query Input Screen

SCHOLARSHIP CANDIDATES/ACADEMIC PERFORMANCE QUERY			
AS Class		Cumulative GPA >=	
Scholarship Category (T, N, P)		AFOQT Quan >=	10
		AFOQT Verb >=	15
Last Name		AFOQT Pilot >=	50
		AFOQT Nav >=	30
		Cumulative SAT	

	Query Item	Operators[<,>,<=>,<=>,<=>]	Query Values
EXAMPLE	Last Name	>= * Absence of Operator < field defaults to '='	ANDERSON SMITH

### Report Formats

#### SCHOLARSHIP CANDIDATES/ACADEMIC PERFORMANCE REPORT

First Name	Last Name	AS Class	Cat Type	GPA Cum	SAT Cum	AFOQT Quan	Verb	Pil	Nav
CARTER	FRANK	3	2	3.50	1200	80	80	80	80

#### SCHOLARSHIP CANDIDATES/ACADEMIC PERFORMANCE REPORT

First Name	Last Name	AS Class	Cat Type	GPA Cum	SAT Cum	AFOQT Quan	Verb	Pil	Nav	AcAp	AFOQT Date	ACT Cum	WPSS Score	AS Class Rank	FY Rating	GPA Sem
CARTER	FRANK	3	2	3.50	1200	80	80	80	80	80	01/12/85	30	103.22	9/ 23	45	3.60

## Query Input Screen

DATE OF COMMISSIONING (DOC) FISCAL YEAR QUERY	
DOC            >= 88	Fiscal Year >= 40
Fiscal Year	Rating
Last Name	Det Commander >= 6
	Rating
SSAN            - -	

	Query Item	Operators[<,>,<=>,<=>,<=>]	Query Values
EXAMPLE	Last Name	>= * Absence of Operator < field defaults to '='	ANDERSON SMITH

## Report Formats

### DATE OF COMMISSIONING (DOC) FISCAL YEAR REPORT

First Name	Last Name	FY Rating	DC Rating	AS Class Rank	AS Class	Comm Date
CARTER	FRANK	45	7	9/ 23	3	01/10/86

### DATE OF COMMISSIONING (DOC) FISCAL YEAR REPORT

First Name	Last Name	FY Rating	DC Rating	AS Class Rank	AS Class	Comm Date	Grad Date	Cat Type	WPSS Score	GPA Cum	SAT Cum	FT Comp	FT Rating
CARTER	FRANK	45	7	9/ 23	3	01/10/86	01/10/86	2	103.22	3.50	1200	N	555.55

## Query Input Screen

AIR SCIENCE CLASS GENERAL INFORMATION QUERY	
AS Class	>= 1 <= 3
Category Type	2
Pursuing/Conditional	C
Last Name	
SSAN	- - -

	Query Item	Operators[<,>,<=>,<=>,>=]	Query Values
EXAMPLE	Last Name	>=	* Absence of Operator
		<	field defaults to '='
			ANDERSON SMITH

## Report Formats

### AIR SCIENCE CLASS GENERAL INFORMATION REPORT

First Name	Last Name	AS Class	Cat Type	Purs Major	Schl Cond	Min Type	Min Math	Min Eng	Min Frl
CARTER	FRANK	3	2	MIS	P	3.0	N	N	N

### AIR SCIENCE CLASS GENERAL INFORMATION REPORT

First Name	Last Name	AS Class	Cat Type	Purs Major	Schl Cond	Min Type	Min Math	Min Eng	Min Frl	SSAN	Matric	Work	Corps	Auxiliaries
CARTER	FRANK	3	2	MIS	P	3.0	N	N	N	111-11-1111	506201	N	AA:SW:	: : : : :

### Query Input Screen

TWO-YEAR PROGRAM CANDIDATE (HORIZONTAL AXIS) QUERY

AS Class = 2

Category Type 3

Last Name

SSAN - -

	Query Item	Operators[<,>=,<>,<=,>=]	Query Values
EXAMPLE	Last Name	>=	* Absence of Operator
		<	field defaults to '=' SMITH

## Report Formats

## TWO-YEAR PROGRAM CANDIDATE (HORIZONTAL AXIS) REPORT

First Name	Last Name				AS Class	Cat Type	Phys Cat	Physical Date		ALTU	Race
CARTER	FRANK				3	2	2	01/03/89		N	C
AFOQT					SAT				GPA		DC
Quan	Verb	Pil	Nav	AcAp	Cum	Math	Verb	Cum	Sem	Rtngr	
80	80	80	80	80	1200	600	600	3.50	3.60	7	

## TWO-YEAR PROGRAM CANDIDATE (HORIZONTAL AXIS) REPORT

First Name	Last Name	AS Class	Cat Type	Phys Cat	Physical Date	ALTU	Race	LOCAL Street	City	Zip	Phone
ARTER	FRANK	3	2	2	01/03/89	N	C	5305 CARRIAGE HILLS	TUCSON	85746	741-0736

AFOQT					SAT			GPA		DC	ACT		Form 48			
Quan	Verb	Pll	Nav	AcAp	Cum	Math	Verb	Cum	Sem	Rtng	Cum	Math	Engl	NSci	SSci	Date
80	80	80	80	80	1200	600	600	3.50	3.60	7	30	30	30	30	30	01/09/87

### Query Input Screen

GRADUATION/COMMISSIONING SUSPENSE DATES QUERY			
AS Class	=	4	# Days Until >= 30 Commissioning Date <= 90
Last Name			# Days Until Graduation Date
SSAN	-	-	

	Query Item	Operators[<,>,<=>,<=,>=]	Query Values
EXAMPLE	Last Name	>= * Absence of Operator < field defaults to '='	ANDERSON SMITH

### Report Formats

#### GRADUATION/COMMISSIONING SUSPENSE DATES REPORT

First Name	Last Name	Comm Date	Grad Date	AS Class	SSAN
CARTER	FRANK	01/10/86	01/10/86	3	111-11-1111

#### GRADUATION/COMMISSIONING SUSPENSE DATES REPORT

First Name	Last Name	Comm Date	Grad Date	AS Class	SSAN
CARTER	FRANK	01/10/86	01/10/86	3	111-11-1111

### Query Input Screen

SCHOLARSHIP EXPIRATION DATES QUERY			
AS Class >= 3  Category Type     2  Scholarship Type     >= 2.0 <= 4.0  Last Name  SSAN                 -   -			

	Query Item	Operators[<,>,<=>,<=,>=]	Query Values
EXAMPLE	Last Name	>=	* Absence of Operator
		<	field defaults to '='
			ANDERSON SMITH

### Report Formats

#### SCHOLARSHIP EXPIRATION DATES REPORT

First Name	Last Name	Schl Exp Date	Sch Typ	Corps Position	Semester Intrview
CARTER	FRANK	15/05/89	3.0	CORPS SGT MAJOR	01/09/87

#### SCHOLARSHIP EXPIRATION DATES REPORT

First Name	Last Name	Schl Exp Date	Sch Typ	Corps Position	Semester Intrview	Significant Information
CARTER	FRANK	15/05/89	3.0	CORPS SGT MAJOR	01/09/87	FATHER-> VICE CMNDR FOR NATO FORCES IN EUROPE



## Query Input Screen

CADET WEIGHT AND AEROBIC STANDARDS QUERY	
AS Class >= 1  Last Name  SSAN       -   -  Print Options *Subject to constraints above* All Cadets - 1 Only Cadets in violation of standards - 2 1	

	Query Item	Operators(<,>,<=,>=)	Query Values
EXAMPLE	Last Name	>= * Absence of Operator < field defaults to '='	ANDERSON SMITH

## Report Formats

### CADET WEIGHT AND AEROBIC STANDARDS REPORT

First	Last			Max	Min	Max	Min		Max
Name	Name	Heigh	Weight	Weight	Weight	WT	WT	10%	RT
CARTER	FRANK	69.25	154.00	190.25	119.00				
		AS	Cat	Run	Max				
		Class	Type	Age	Time	Run	Time		
		3	2	28	8:30	12:00			

### CADET WEIGHT AND AEROBIC STANDARDS REPORT

First Name	Last Name	Heigh	Weight	Max Weight	Min Weight	Max WT	Min WT	10%	RT	Max	LOCAL Street	City	Zip	Phone
CARTER	FRANK	69.25	154.00	190.25	119.00						5365 CARRIAGE HILLS	TUCSON	85746	741-0736
		AS	Cat	Run	Max									
		Class	Type	Age	Time	Run Time								
		3	2	28	8:30	12:00								

# INDIVIDUAL CADET QUERY

Enter Name or Social Security #

First Name  
Middle Name  
Last Name

SSAN - -

Query Input  
Screen

## INDIVIDUAL CADET REPORT (Press any key to continue)

First Name	Middle Name	Last Name	SSAN	Matric	Birth Date	Age	Sex
CARTER	LEROY	FRANK	111-11-1111	506291	05/10/58	28	M

AS Yr	AS Class	DC Rtn	FY Rtn	FT Rating	FT Cmp	Pil	Corps
3	10/1	7	45	555.55	N	N	AA SW

Cat Type	Purs Cond	4-Yr Cad.	Pri Serv	Waiv Req	Form 48 Date	Semester Interview	FSP Date
2	P	N	N	N	01/09/87	01/09/87	C / /

Report Format

Height	Weight	Weigh Date	Run Time	Run Date	Phys Cat	Phys Date	Grad Date	Comm Date
69.25	154.00	10/10/86	8:30	10/10/86	2	01/03/89	01/10/86	01/10/86

Major	Schl Type	Schl Exp Date	GPA Cum	Sem Cum	SAT Cum	Math	Verb	ACT Cum	Math	Engl	NSci	SSci
MIS	3.0	15/05/89	3.50	3.60	1200	600	600	30	30	30	30	30

AFOQT Quan	Verb	Pil	Nav	AcAp	AFOQT Date	Min Req Math	Engl	Frln
80	80	80	80	80	01/12/85	N	N	N

# INDIVIDUAL CADET PAY QUERY

Enter Name or Social Security #

First Name  
Middle Name  
Last Name

SSAN 111-11-1111

Query Input  
Screen

## INDIVIDUAL CADET PAY REPORT (Press any key to continue)

First Name	Middle Name	Last Name	SSAN	Matric	AS Class	Cat Type	Schl Type
CARTER	LEROY	FRANK	111-11-1111	506291	3	2	3.0

Pay Period	Start Pay Date	Stop Pay Date	Res Stat	Tuition	Book Fees	FT Days	ATP Days	FSP Days	Num Days	Cum Days
1	01/09/85	31/12/85	0	1300.00	100.00	0	0	0	122	122
2	01/01/86	31/05/86	I	600.00	150.00	0	0	0	151	273
3	01/09/86	31/12/86	I	700.00	200.00	0	0	0	122	395
4	01/01/87	31/05/87	I	800.00	250.00	0	0	0	151	546
5	01/06/87	31/08/87		0.00	0.00	28	14	14	36	582
6	01/09/87	31/12/87	I	900.00	300.00	0	0	0	122	704
7	01/01/88	31/05/88	I	1000.00	350.00	0	0	0	152	856
8	01/09/88	31/12/88	I	1100.00	425.00	0	0	0	122	978
9	01/01/89	31/05/89	I	1200.00	450.00	0	0	0	151	1129
10	01/06/89	31/08/89	I	750.00	175.00	0	0	0	92	1221
(Column Totals)-->				8350.00	2400.00	28	14	14		

Report Format

RCIS UTILITIES

Version 1.10  
Copyright (C) 1987  
by  
Carter L. Frank  
All rights reserved

Utilities  
Log-on Screen

RCIS UTILITIES

FUNCTION

BackUp  
ReLoad  
PassWord  
Done

Backup  
Selection  
Response

Insert a formatted disk in drive A and press any key.

RCIS UTILITIES

FUNCTION

BackUp  
ReLoad  
PassWord  
Done

Reload  
Selection  
Response

WARNING: This option will erase existing files.

Do you want to continue? N

RCIS UTILITIES

FUNCTION

BackUp  
ReLoad  
PassWord  
Done

Password  
Selection  
Response

Enter old password

Enter new password

Verify new password

TECHNICAL MANUAL  
FOR  
ROTC CADET INFORMATION SYSTEM ( RCIS )  
VERSION 1.10

BY  
Carter L. Frank

A Report Submitted in Partial Fulfillment of the  
Requirements for the Degree of Master of Science  
(Management Information Systems)  
in The University of Arizona

1987

Master Committee:  
Dr. Sudha Ram

## TABLE OF CONTENTS

Title page.....	i
Table of Contents.....	ii
Table of Figures.....	iii
1.0 Introduction.....	1
1.1 Overview.....	1
1.2 RCIS Requirements.....	1
2.0 Overview.....	2
2.1 Data Dictionary.....	2
2.2 Entity Relationship Diagram.....	15
2.3 Normalization.....	16
2.4 Database Structures for RCIS.....	19

## TABLE OF FIGURES

2.1 RCIS Environment.....	15
---------------------------	----

## 1.0 INTRODUCTION.

This manual provides technical information for the ROTC Cadet Information System (RCIS) database and program source code. Section 2 focuses on the design of the database. Attachment 1 contains a copy of the documented source code for the program.

### 1.1 OVERVIEW.

Section 2 provides information used to ultimately design the relations contained in the RCIS database. The section documents the activities in all four phases of the database design. Materials contained in this section include:

- a. Data dictionary of attributes contained in the database.
- b. Entity Relationship Diagram of the database.
- c. Functional dependencies used during normalization.
- d. Final relational schema and indices.

### 1.2 RCIS REQUIREMENTS.

RCIS was designed to be run on an IBM PC/AT or compatible under dBASE III PLUS, Version 1.1. The minimum hardware requirements for the system include:

- a. 512K RAM.
- b. Monochrome monitor.
- c. One floppy disk drive.
- d. One hard disk drive.

## 2.0 OVERVIEW.

This section contains documentation of the database design phases including: a data dictionary of all the attributes contained in RCIS relations, an Entity Relationship Model (ERM) of the RCIS environment, functional dependencies used to decompose and normalize the relational schema, and the final relational schemata.

### 2.1 DATA DICTIONARY.

AA_NUM	Type : Numeric Width : 6 Dec: 4 Format : 9.9999 Remarks : Numeric value multiplied by cadet's AFOQT Academic Aptitude score in figuring the WPSS score.
ACT_CUM	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's cumulative ACT score.
ACT_ENGL	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's ACT english score.
ACT_MATH	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's ACT math score.
ACT_NSCI	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's ACT natural science score.



ACT_SSCI	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's ACT social science score.
AFOQT_AA	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's AFOQT academic aptitude score.
AFOQT_DATE	Type : Date Width : 8 Format : 99/99/9999 Remarks : The cadet's AFOQT test date.
AFOQT_NAV	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's AFOQT navigator score.
AFOQT_PLT	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's AFOQT pilot score.
AFOQT_QUAN	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's AFOQT quantitative score.
AFOQT_VERB	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's AFOQT verbal score.
AGE	Type : Character Width : 2 Format : 99 Remarks : The cadet's age.
AGE_CAT	Type : Character Width : 1 Format : 9 Remarks : The cadet's age category ( 1 for < 30 yrs, 2 otherwise).

ALTU	Type : Logical Width : 1 Format : Y/N Remarks : Indicates whether the cadet has completed the mock field training camp.
AS_CLASS	Type : Numeric Width : 1 Format : 9 Remarks : The aerospace class the cadet is enrolled in (1 = freshman, 2 = sophomore, 3 = junior, 4 = senior).
AS_CL_TOT	Type : Numeric Width : 3 Format : 999 Remarks : The total number of cadets enrolled in each specific aerospace class.
AS_RNK_POS	Type : Numeric Width : 3 Format : 999 Remarks : Aerospace studies ranking of the cadet in each aerospace studies class.
ATP_DAYS	Type : Numeric Width : 2 Format : 99 Remarks : Number of days the cadet attended a pilot training school.
BIRTHDATE	Type : Date Width : 8 Format : 99/99/9999 Remarks : Cadet's birth date.
BOOK_FEES	Type : Numeric Width : 6 Dec: 2 Format : 999.99 Remarks : Cadet's expenses for books/notes during a specified period of time (for contract cadets only).

CAT\_TYPE      Type      : Character  
                  Width     : 1  
                  Format    : PIC X  
                  Remarks : Code representing the cadet's category  
                          type (M = missile, N = navigator, P =  
                          pilot, Q = nurse, R = pre-med, L = law  
                          2 = technical, 3 = non-technical).

COM\_DATE      Type      : Date  
                  Width     : 8  
                  Format    : 99/99/9999  
                  Remarks : Cadet's commissioning date.

CORPS\_AUX     Type      : Character  
                  Width     : 16  
                  Format    : XX/XX/XX/XX/XX/XX/XX/XX  
                  Remarks : Two-digit codes indicating the cadet's  
                          participation in corps auxiliaries.

CORPS\_POS     Type      : Character  
                  Width     : 25  
                  Format    : PIC X(25)  
                  Remarks : The cadet's assigned position in the  
                          corps.

CUM\_GPA       Type      : Numeric  
                  Width     : 4    Dec: 2  
                  Format    : 9.99  
                  Remarks : The cadet's cumulative GPA.

DCR\_NUM       Type      : Numeric  
                  Width     : 6    Dec: 4  
                  Format    : 9.9999  
                  Remarks : Numeric value multiplied by cadet's  
                          Detachment    Commander    rating    in  
                          figuring the WPSS score.

DC\_RATING     Type      : Numeric  
                  Width     : 1  
                  Format    : 9  
                  Remarks : The Detachment Commander's rating of  
                          each cadet.

FORM\_48      Type      : Date  
              Width     : 8  
              Format    : 99/99/9999  
              Remarks  : Last completion date for the cadet's  
                       most current Air Force Form 48 (degree  
                       plan).

FOUR\_YR      Type      : Logical  
              Width     : 1  
              Format    : Y/N  
              Remarks  : Indicates whether the cadet is a four  
                       year AFROTC student.

FSP\_DATE     Type      : Date  
              Width     : 8  
              Format    : 99/99/9999  
              Remarks  : Flight screening program completion  
                       date (program for potential pilot  
                       cadets).

FSP\_DAYS     Type      : Numeric  
              Width     : 2  
              Format    : 99  
              Remarks  : Number of days the cadet attended  
                       the flight screening program.

FT\_COMP      Type      : Logical  
              Width     : 1  
              Format    : Y/N  
              Remarks  : Indicates whether the cadet has  
                       completed field training.

FT\_DAYS      Type      : Numeric  
              Width     : 2  
              Format    : 99  
              Remarks  : Number of days the cadet attended  
                       field training.

FT\_RTNG      Type      : Numeric  
              Width     : 6   Dec: 2  
              Format    : 999.99  
              Remarks  : Advisor's rating of the cadet's  
                       performance at field training.

FY_RTNG	Type : Numeric Width : 2 Format : 99 Remarks : The cadet's fiscal year rating score.
F_NAME	Type : Character Width : 15 Format : PIC X(15) Remarks : The cadet's first name.
GPA_NUM	Type : Numeric Width : 6 Dec: 4 Format : 9.9999 Remarks : Numeric value multiplied by cadet's cumulative GPA in figuring the WPSS score.
GRAD_DATE	Type : Date Width : 8 Format : 99/99/9999 Remarks : The cadet's graduation date.
HEIGHT	Type : Numeric Width : 5 Dec: 2 Format : 99.99 Remarks : The cadet's height in inches and quarter inches.
LOCAL_CITY	Type : Character Width : 20 Format : PIC X(20) Remarks : City name associated with the cadet's local address.
LOCAL_PHON	Type : Character Width : 7 Format : 999-9999 Remarks : Cadet's local phone number.
LOCAL_STRT	Type : Character Width : 30 Format : PIC X(30) Remarks : Street name associated with cadet's local address.

LOCAL\_ZIP      Type        : Character  
                  Width       : 9  
                  Format      : 99999-XXXX  
                  Remarks    : Zipcode associated with cadet's local  
                                  address.

L\_NAME           Type        : Character  
                  Width       : 15  
                  Format      : PIC X(15)  
                  Remarks    : Cadet's last name.

MAJOR            Type        : Character  
                  Width       : 4  
                  Format      : PIC X(4)  
                  Remarks    : Four-character code for the cadet's  
                                  academic major.

MATRIC           Type        : Character  
                  Width       : 6  
                  Format      : 999999  
                  Remarks    : The cadet's six-digit matriculation  
                                  number.

MAX\_RT\_F         Type        : Numeric  
                  Width       : 4  
                  Format      : 9999  
                  Remarks    : Maximum allowable time for a female  
                                  cadet to run a mile and a half.

MAX\_RT\_M         Type        : Numeric  
                  Width       : 4  
                  Format      : 9999  
                  Remarks    : Maximum allowable time for a male  
                                  cadet to run a mile and a half.

MAX\_WT\_F         Type        : Numeric  
                  Width       : 6    Dec: 2  
                  Format      : 999.99  
                  Remarks    : Maximum allowable weight for a female  
                                  cadet at her measured height.

MAX\_WT\_M         Type        : Numeric  
                  Width       : 6    Dec: 2  
                  Format      : 999.99  
                  Remarks    : Maximum allowable weight for a male  
                                  cadet at his measured height.

MIN_WT_F	Type : Numeric Width : 6 Dec: 2 Format : 999.99 Remarks : Minimum allowable weight for a female cadet at her measured height.
MIN_WT_M	Type : Numeric Width : 6 Dec: 2 Format : 999.99 Remarks : Minimum allowable weight for a male cadet at his measured height.
M_NAME	Type : Character Width : 15 Format : PIC X(15) Remarks : The cadet's middle name.
M_R_ENGL	Type : Logical Width : 1 Format : Y/N Remarks : Indicates whether the cadet has completed the minimum required english courses.
M_R_FLAN	Type : Logical Width : 1 Format : Y/N Remarks : Indicates whether the cadet has completed the minimum required foreign language courses.
M_R_MATH	Type : Logical Width : 1 Format : Y/N Remarks : Indicates whether the cadet has completed the minimum required math courses.
OTHER_INFO	Type : Character Width : 50 Format : PIC X(50) Remarks : Significant information about the cadet, i.e. cadet's father is a general.

PAY\_DATE1    Type    : Date  
              Width    : 8  
              Format   : 99/99/9999  
              Remarks : Beginning date for a pay period.

PAY\_DATE2    Type    : Date  
              Width    : 8  
              Format   : 99/99/9999  
              Remarks : Ending date for a pay period.

PERM\_CITY    Type    : Character  
              Width    : 20  
              Format   : PIC X(20)  
              Remarks : City name associated with the cadet's  
                          permanent address.

PERM\_PHON    Type    : Character  
              Width    : 10  
              Format   : (999)999-9999  
              Remarks : Cadet's permanent phone number.

PERM\_STAT    Type    : Character  
              Width    : 2  
              Format   : PIC X(2)  
              Remarks : State associated with cadet's  
                          permanent address.

PERM\_STRT    Type    : Character  
              Width    : 30  
              Format   : PIC X(30)  
              Remarks : Street name associated with cadet's  
                          local address.

PERM\_ZIP    Type    : Character  
              Width    : 9  
              Format   : 99999-XXXX  
              Remarks : Zipcode associated with cadet's  
                          permanent address.

PHY\_CAT    Type    : Character  
              Width    : 1  
              Format   : PIC X  
              Remarks : The cadet's physical category type.



PHY\_DATE      Type        : Date  
                  Width     : 8  
                  Format    : 99/99/9999  
                  Remarks : The date of the cadet's physical  
                                  qualification examination.

PLT\_LICENS    Type        : Logical  
                  Width     : 1  
                  Format    : Y/N  
                  Remarks : Indicates whether the cadet has a  
                                  private pilot's license.

PRIOR\_SVC     Type        : Logical  
                  Width     : 1  
                  Format    : Y/N  
                  Remarks : Indicates whether the cadet has had  
                                  prior military service experience.

PC\_STATUS     Type        : Character  
                  Width     : 1  
                  Format    : PIC X  
                  Remarks : Code indicating whether the cadet is  
                                  on pursuing [P] or conditional [S]  
                                  status.

QUAN\_NUM       Type        : Numeric  
                  Width     : 6    Dec: 4  
                  Format    : 9.9999  
                  Remarks : Numeric value multiplied by cadet's  
                                  AFOQT quantitative score in figuring  
                                  the WPSS score.

RACE           Type        : Character  
                  Width     : 1  
                  Format    : PIC X  
                  Remarks : Code for cadet's race.

RES\_STATUS    Type        : Character  
                  Width     : 1  
                  Format    : PIC X  
                  Remarks : Code for cadet's residency status, [I]  
                                  for in-state, [O] for out-of-state.

RUN\_DATE      Type     : Date  
              Width    : 8  
              Format   : 99/99/9999  
              Remarks : Date of the cadet's aerobics run time.

RUN\_TIME      Type     : Character  
              Width    : 4  
              Format   : 9999  
              Remarks : The cadet's aerobics run time (first  
                       two digits are minutes, second two  
                       digits are seconds).

SAT\_CUM       Type     : Numeric  
              Width    : 2  
              Format   : 99  
              Remarks : The cadet's cumulative SAT score.

SAT\_MATH      Type     : Numeric  
              Width    : 2  
              Format   : 99  
              Remarks : The cadet's SAT math score.

SAT\_NUM       Type     : Numeric  
              Width    : 6   Dec: 4  
              Format   : 9.9999  
              Remarks : Numeric value multiplied by cadet's  
                       cumulative SAT score in figuring the  
                       WPSS score.

SAT\_VERB      Type     : Numeric  
              Width    : 2  
              Format   : 99  
              Remarks : The cadet's SAT verbal score.

SCHLR\_DATE    Type     : Date  
              Width    : 8  
              Format   : 99/99/9999  
              Remarks : The expiration date of the cadet's  
                       ROTC scholarship.

SCHLR\_TYPE    Type     : Numeric  
              Width    : 3   Dec: 1  
              Format   : 9.9  
              Remarks : The cadet's AFROTC scholarship type,  
                       2.5 = two and a half year scholarship.

SEM_GPA	Type : Numeric Width : 4 Dec: 2 Format : 9.99 Remarks : The cadet's most current semester GPA.
SEM_INTRVW	Type : Date Width : 8 Format : 99/99/9999 Remarks : Date of the cadet's most recent semester interview.
SEX	Type : Character Width : 1 Format : PIC X Remarks : The cadet's gender.
SSAN	Type : Character Width : 9 Format : 999-99-9999 Remarks : The cadet's social security number.
TUITION	Type : Numeric Width : 7 Dec: 2 Format : 9999.99 Remarks : The cadet's tuition for a given semester (in dollars and cents).
VERB_NUM	Type : Numeric Width : 6 Dec: 4 Format : 9.9999 Remarks : Numeric value multiplied by cadet's AFOQT verbal score in figuring the WPSS score.
WAIVER_REQ	Type : Logical Width : 1 Format : Y/N Remarks : Indicates whether the cadets has a waiver required on their physical.
WEIGHT	Type : Numeric Width : 6 Dec: 2 Format : 999.99 Remarks : The cadet's weight in pounds and quarter pounds.

WEIGH\_DATE    Type    : Date  
                  Width    : 8  
                  Format   : 99/99/9999  
                  Remarks : Date the cadet's weight was measured.

WORK            Type    : Logical  
                  Width    : 1  
                  Format   : Y/N  
                  Remarks : Indicates whether the cadet has a  
                              parttime job.

WPSS            Type    : Numeric  
                  Width    : 6    Dec: 2  
                  Format   : 999.99  
                  Remarks : Numerical score calculated using the  
                              following data:    DC\_RTNG,    DCR\_NUM,  
                              CUM\_GPA,    GPA\_NUM,    SAT\_CUM,    SAT\_NUM,  
                              AFOQT\_AA,    AA\_NUM,    AFOQT\_QUAN,    QUAN\_NUM  
                              AFOQT\_VERB,    VERB\_NUM

## 2.2 ENTITY RELATIONSHIP DIAGRAM.

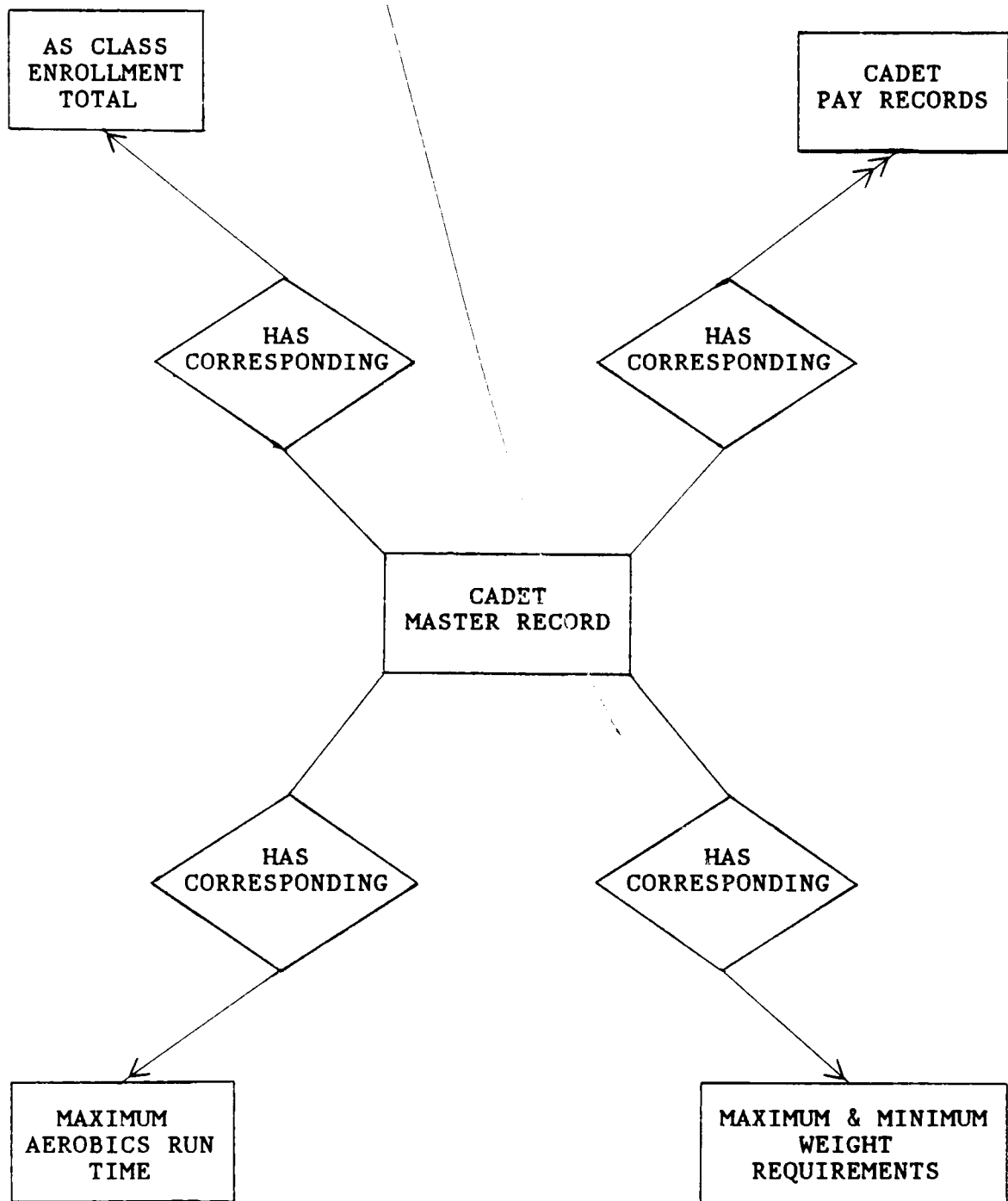


Figure 2.1 RCIS Environment

## 2.3 NORMALIZATION.

This section presents the functional dependencies (FDs) used to normalize RCIS relations by the decomposition approach. The notation  $X \twoheadrightarrow Y$  is used to indicate a functional relationship between the attribute  $X$  and  $Y$ . The notation  $X \twoheadrightarrow\!\!\rightarrow Y$  is used to denote a multivalued dependency.

### 2.3.1 A\_CDT\_MS.DBF AND I\_CDT\_MS.DBF

SSAN $\twoheadrightarrow$	ACT_CUM	SSAN $\twoheadrightarrow$	L_NAME
	ACT_ENGL		MAJOR
	ACT_MATH		MATRIC
	ACT_NSCI		M_NAME
	ACT_SSCI		M_R_ENGL
	AFOQT_AA		M_R_FLAN
	AFOQT_DATE		M_R_MATH
	AFOQT_NAV		OTHER_INFO
	AFOQT_PLT		PERM_CITY
	AFOQT_QUAN		PERM_PHON
	AFOQT_VERB		PERM_STAT
	AGE		PERM_STRT
	ALTU		PERM_ZIP
	AS_CLASS		PHY_CAT
	AS_RNK_POS		PHY_DATE
	BIRTHDATE		PLT_LICENS
	CAT_TYPE		PRIOR_SVC
	COM_DATE		PC_STATUS
	CORPS_AUX		RACE
	CORPS_POS		RUN_DATE
	CUM_GPA		RUN_TIME
	DC_RTNG		SAT_CUM
	FORM_48		SAT_MATH
	FOUR_YR		SAT_VERB
	FSP_DATE		SCHLR_DATE
	FT_COMP		SCHLR_TYPE
	FT_RTNG		SEM_GPA
	FY_RTNG		SEM_INTRVW
	F_NAME		SEX
	GRAD_DATE		WAIVER_REQ
	HEIGHT		WEIGHT
	LOCAL_CITY		WEIGH_DATE
	LOCAL_PHON		WORK
	LOCAL_STRT		WPSS
	LOCAL_ZIP		

ANALYSIS: This relation is in 4NF. See the RCIS User's Guide for a discussion of the indices created to support this relation.

### 2.3.2 A\_CDT\_PY.DBF AND I\_CDT\_PY.DBF

SSAN, PAY\_DATE1 --> ATP\_DAYS  
BOOK\_FEES  
FSP\_DAYS  
FT\_DAYS  
PAY\_DATE2  
RES\_STATUS  
TUITION

ANALYSIS: This relation is in 4NF. See the RCIS User's Guide for a discussion of the indices created to support this relation.

### 2.3.3 A\_CDT\_CT.DBF AND I\_CDT\_CT.DBF

AS\_CLASS --> AS\_CL\_TOT

ANALYSIS: This relation is in 4NF. See the RCIS User's Guide for a discussion of the indices created to support this relation.

### 2.3.4 T\_CDT\_RT.DBF

AGE\_CAT --> MAX\_RT\_F  
MAX\_RT\_M

ANALYSIS: This table is in 4NF. The AGE\_CAT field is determined inside the program source code by using the AGE field of the relation in section 2.3.1 (AGE\_CAT = 1 when AGE < 30; AGE\_CAT = 2 when AGE >= 30). See the RCIS User's Guide for a discussion of the indices created to support this relation.

### 2.3.5 T\_CDT\_HW.DBF

HEIGHT --> MAX\_WT\_F  
MAX\_WT\_M  
MIN\_WT\_F  
MIN\_WT\_M

ANALYSIS: This table is in 4NF. The appropriate MAX\_WT and MIN\_WT are determined inside the program source code by using the SEX field of the relation in section 2.3.1 (SEX = 'F' then use MAX\_WT\_F and MIN\_WT\_F; SEX = 'M' then use MAX\_WT\_M and MIN\_WT\_M). The following FDs make up an alternate design for this table:

SEX, HEIGHT --> MAX\_WT  
MIN\_WT

This design would give us a relation of less degree (lower number of columns) but it would double the cardinality (twice as many rows). The decision not to use this design was based on the idea that a micro-based system normally has limited processing capabilities therefore smaller files are processed faster. See the RCIS User's Guide for a discussion of the indices created to support this relation.

### 2.3.6 T\_CDT\_WP.DBF

AA\_NUM  
DCR\_NUM  
GPA\_NUM  
QUAN\_NUM  
SAT\_NUM  
VERB\_NUM

ANALYSIS: This table is in is not in any normal form since it has no key and is merely a convenient storage location for this one record of WPSS multiplier values.



## 2.4 DATABASE STRUCTURES FOR RCIS.

This section presents the final RCIS relations and identifies the primary and secondary access keys. The primary key is denoted by the symbol "\$" and the secondary keys are indicated by the symbol "\*". The number of bytes/record for each relation is also presented.

### 2.4.1 A\_CDT\_MS.DBF AND I\_CDT\_MS.DBF

474 bytes/record

\$ SSAN	LOCAL_ZIP
ACT_CUM	* L_NAME
ACT_ENGL	MAJOR
ACT_MATH	MATRIC
ACT_NSCI	* M_NAME
ACT_SSCI	M_R_ENGL
AFOQT_AA	M_R_FLAN
AFOQT_DATE	M_R_MATH
AFOQT_NAV	OTHER_INFO
AFOQT_PLT	PERM_CITY
AFOQT_QUAN	PERM_PHON
AFOQT_VERB	PERM_STAT
AGE	PERM_STRT
ALTU	PERM_ZIP
AS_CLASS	PHY_CAT
AS_RNK_POS	PHY_DATE
BIRTHDATE	PLT_LICENS
CAT_TYPE	PRIOR_SVC
COM_DATE	PC_STATUS
CORPS_AUX	RACE
CORPS_POS	RUN_DATE
CUM_GPA	RUN_TIME
DC_RTNG	SAT_CUM
FORM_48	SAT_MATH
FOUR_YR	SAT_VERB
FSP_DATE	SCHLR_DATE
FT_COMP	SCHLR_TYPE
FT_RTNG	SEM_GPA
FY_RTNG	SEM_INTRVW
* F_NAME	SEX
GRAD_DATE	WAIVER_REQ
HEIGHT	WEIGHT
LOCAL_CITY	WEIGH_DATE
LOCAL_PHON	WORK
LOCAL_STRT	WPSS

2.4.2 A\_CDT\_PY.DBF AND I\_CDT\_PY.DBF

46 bytes/record

\$ SSAN, PAY\_DATE1 (Composite primary key)  
ATP\_DAYS  
BOOK\_FEES  
FSP\_DAYS  
FT\_DAYS  
PAY\_DATE2  
RES\_STATUS  
TUITION

2.4.3 A\_CDT\_CT.DBF AND I\_CDT\_CT.DBF

5 bytes/record

\$ AS\_CLASS  
AS\_CL\_TOT

2.4.4 T\_CDT\_RT.DBF

10 bytes/record

\$ AGE\_CAT  
MAX\_RT\_F  
MAX\_RT\_M

2.4.5 T\_CDT\_HW.DBF

30 bytes/record

\$ HEIGHT  
MAX\_WT\_F  
MAX\_WT\_M  
MIN\_WT\_F  
MIN\_WT\_M

2.4.6 T\_CDT\_WP.DBF

37 bytes/record

AA\_NUM  
DCR\_NUM  
GPA\_NUM  
SAT\_NUM  
QUAN\_NUM  
VERB\_NUM

ATTACHMENT 1  
(SOURCE CODE LISTING)

# SOURCE CODE LISTING

## TABLE OF CONTENTS

RCIS.PRG.....1

### RCIS\_P1.PRG

INIT.....	15
SET_MENU.....	16
BOX_CHAR.....	17
F_MENU.....	19
G_MENU.....	22
R_MENU.....	23
QS_MENU.....	24
QO_MENU.....	25

### RCIS\_P2.PRG

ADD_REC.....	26
ADD_PAY.....	31
EDIT_REC.....	35
EDIT_SSAN.....	40
EDIT_PAY.....	43
EDT_LINE.....	46
ED_GETS.....	51
DEL_REC.....	53
DEL_PAY.....	58
DEL_FLGS.....	60
VIEW_REC.....	63
TRANS_REC.....	67
TRANS_CHK.....	73
HGHT_CHK.....	75
VIEW_PAY.....	77
SET_UP.....	80
INPUT_KEY.....	82
SSAN_CHK.....	85
INIT_DB.....	86
BLD_NDX.....	88
INIT_SAV.....	90
INIT_FLG.....	91
SAV_RECS.....	92
RCIS_HDR.....	94
ERR_RE.....	95
ERR_NF.....	96
P_PROMPT.....	98
M_PROMPT.....	99
D_PROMPT.....	100
TQ_PRMP.....	101
VP_PRMP.....	102
DB3_ERR.....	103

SOURCE CODE LISTING  
TABLE OF CONTENTS, CONTINUED

RCIS\_P3.PRG

QUERIES.....	106
WPSS_QRY.....	110
SCHA_QRY.....	122
DCFY_QRY.....	132
CLAS_QRY.....	141
HRAK_QRY.....	149
CGDT_QRY.....	157
SEDT_QRY.....	165
WTAR_QRY.....	173
INDV_QRY.....	182
PAYI_QRY.....	192
HELP_SCRN.....	200
ERR_NF.....	201
RCIS_HDR.....	202
M_PROMPT.....	203
RO_CHK.....	204
SET_DB.....	205
DB3_Q_ERR.....	207

RCISUTIL.PRG.....	210
-------------------	-----

RCISUTL2.PRG

U_INIT.....	212
CHK_NDX.....	214
CHK_DSK.....	216
SET_DSK.....	217
LOAD_DBF.....	219
COPY_DBF.....	221
UBACKUP.....	225
URELOAD.....	227
PASSWORD.....	229

MENU.ASM.....	231
---------------	-----

CDT_M.FMT.....	238
----------------	-----

CDT_M_VU.FMT.....	241
-------------------	-----

```

*-----*
*               BEGINNING OF RCIS.PRG               *
*-----*
* SUMMARY:
*   RCIS.PRG is the main driver for the ROTC Cadet Information System
*   (RCIS) developed for the executive and administrative staff at the
*   AFROTC Detachment 020, University of Arizona. This module ini-
*   tializes program variables, activates pop-up menus to determine
*   user processing requirements, and invokes procedures to add, edit,
*   delete, or transfer records. In addition, this module invokes the
*   query facilities that allow the user to specify ad hoc database
*   queries using form-like query input screens.
*
* CALLED PROCEDURES:
*
*               Procedure Name           Location
*               -----
*               INIT                     RCIS_P1.PRG
*               MENU                     MENU.BIN
*               ADD_REC                  RCIS_P2.PRG
*               EDIT_REC                 RCIS_P2.PRG
*               VIEW_REC                 RCIS_P2.PRG
*               DEL_REC                  RCIS_P2.PRG
*               TRANS_REC                RCIS_P2.PRG
*               QUERIES                  RCIS_P3.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name   Status   Purpose
*   -----
*   F_PARA          GLOBAL   Parameter for MENU.BIN that passes pop-up*
*                           function menu descriptions and returns
*                           with user selection. A more detailed
*                           discussion of this parameter is provided
*                           in RCIS_P1.PRG.
*
*   G_PARA          GLOBAL   Parameter for MENU.BIN that passes pop-up*
*                           group menu descriptions and returns with
*                           user selection of active or inactive data*
*                           base. A more detailed discussion of this*
*                           parameter is provided in RCIS_P1.PRG.
*
*   R_PARA          GLOBAL   Parameter for MENU.BIN that passes pop-up*
*                           record menu descriptions and returns with*
*                           user database selection. A more detailed*
*                           discussion of this parameter is provided
*                           in RCIS_P1.PRG.
*
*   QS_PARA         GLOBAL   Parameter for MENU.BIN that passes pop-up*
*                           query selection menu descriptions and re-
*                           turns with user selection. A more de-
*                           tailed discussion of this parameter is
*                           provided in RCIS_P1.PRG.
*
*   QO_PARA         GLOBAL   Parameter for MENU.BIN that passes pop-up*

```

*			query output menu descriptions and re-	*
*			turns with user selection. A more de-	*
*			tailed discussion of this parameter is	*
*			provided in RCIS_P1.PRG.	*
*				*
*	F_SELECT	GLOBAL	Holds the character indicating the func-	*
*			tion selected by the user.	*
*				*
*	G_SELECT	GLOBAL	Holds the character indicating the rela-	*
*			tion selected by the user.	*
*				*
*	R_SELECT	GLOBAL	Holds the character indicating the group	*
*			(active or inactive) selected by the user	*
*				*
*	QS_SELECT	GLOBAL	Holds the character indicating the query	*
*			type selected by the user.	*
*				*
*	QO_SELECT	GLOBAL	Holds the character indicating the output	*
*			media selected by the user.	*
*				*
*	QUIT_KEY	GLOBAL	Boolean variable that is set to TRUE if	*
*			the user either enters a null string or	*
*			presses the <Esc> key when prompted for	*
*			an access key. If the variable is set to	*
*			TRUE, the system discontinues processing	*
*			the current function and returns to the	*
*			main menu.	*
*				*
*	M_CHOICE	GLOBAL	Boolean variable used to flag desire to	*
*			continue with a selected processing mode.	*
*				*
*	P_CHOICE	GLOBAL	Boolean variable used to flag desire to	*
*			add additional Pay records to the	*
*			selected Master record.	*
*				*
*	VP_CHOICE	GLOBAL	Boolean variable used to flag desire to	*
*			view all associated Pay records for the	*
*			selected Master record.	*
*				*
*	TQ_CHOICE	GLOBAL	Boolean variable used to flag desire to	*
*			transfer records from active to inactive	*
*			files or vice versa.	*
*				*
*	FILT_STR	GLOBAL	String variable used to hold filter cond-	*
*			itions required to properly locate the	*
*			desired records.	*
*				*
*	T_FOR_STR	GLOBAL	String variable used to hold secondary	*
*			filter conditions (in this case, name	*
*			variables only) required to properly lo-	*
*			cate the desired records.	*
*				*
*	EMPTY_M	GLOBAL	Boolean variable used to flag the condi-	*
*			tion of an empty Master file.	*
*				*



*	EMPTY_P	GLOBAL	Boolean variable used to flag the condi-	*
*			tion of an empty Pay file.	*
*				*
*	REC_NUM	GLOBAL	Used to store the database system record	*
*			number for the record currently being	*
*			processed.	*
*				*
*	DEL_FLAG	GLOBAL	Boolean variable set to TRUE when the	*
*			current Master record has been marked for	*
*			deletion.	*
*				*
*	FIRST_TIME	GLOBAL	Boolean variable used in many procedures	*
*			when the first pass through a code seg-	*
*			ment requires some "first time" initial-	*
*			izations.	*
*				*
*	IN_SSAN	GLOBAL	Used as a holding area for the primary	*
*			key input by the user.	*
*				*
*	IN_FNAM	GLOBAL	Used as a holding area for one of the	*
*			secondary keys input by the user.	*
*				*
*	IN_MNAM	GLOBAL	Used as a holding area for one of the	*
*			secondary keys input by the user.	*
*				*
*	IN_LNAM	GLOBAL	Used as a holding area for one of the	*
*			secondary keys input by the user.	*
*				*
*	T_PATH	GLOBAL	Used to store a code value indicating	*
*			whether the user would like to try an-	*
*			other record transfer or exit back to	*
*			the function select menu.	*
*				*
*	NDX_LIST	GLOBAL	String variable used to store the list of	*
*			index files that will be updated whenever	*
*			a record is added or deleted.	*
*				*
*	M_FILE	GLOBAL	String variable used to store the name of	*
*			the Master file being used (no extension)	*
*				*
*	P_FILE	GLOBAL	String variable used to store the name of	*
*			the Pay file being used (no extension).	*
*				*
*	CT_FILE	GLOBAL	String variable used to store the name of	*
*			the Enrollment totals file being used	*
*			(no extension).	*
*				*
*	M_NDX	GLOBAL	String variable used to store the name of	*
*			the primary index file for the Master	*
*			file (no extension).	*
*				*
*	P_NDX	GLOBAL	String variable used to store the name of	*
*			the primary index file for the Pay file	*
*			(no extension).	*
*				*

*	CT_NDX	GLOBAL	String variable used to store the name of*
*			the primary index file for the Enrollment*
*			totals file (no extension).
*			*
*	M_NDX_F	GLOBAL	String variable used to store the name of*
*			the primary index file for the Master
*			file (with extension).
*			*
*	P_NDX_F	GLOBAL	String variable used to store the name of*
*			the primary index file for the Pay file
*			(with extension).
*			*
*	CT_NDX_F	GLOBAL	String variable used to store the name of*
*			the primary index file for the Enrollment*
*			totals file (with extension).
*			*
*	M_NDX_STR	GLOBAL	String variable used to hold names of the*
*			database variables to key on when the
*			Master file index is "set".
*			*
*	P_NDX_STR	GLOBAL	String variable used to hold names of the*
*			database variables to key on when the
*			Pay file index is "set".
*			*
*	M_FORM_STR	GLOBAL	String variable used to hold the name of
*			format files to be displayed when the
*			full screen edit commands are issued.
*			*
*	DEST_FILE	GLOBAL	String variable used to hold the text
*			name of the target file (active or
*			inactive).
*			*
*	T_M_FILE	GLOBAL	String variable used to store the name of*
*			the target Master file being used (no
*			extension)
*			*
*	T_P_FILE	GLOBAL	String variable used to store the name of*
*			the target Pay file being used (no
*			extension).
*			*
*	T_CT_FILE	GLOBAL	String variable used to store the name of*
*			the target Enrollment totals file being
*			used (no extension).
*			*
*	T_M_NDX	GLOBAL	String variable used to store the name of*
*			the primary index file for the target
*			Master file (no extension).
*			*
*	T_P_NDX	GLOBAL	String variable used to store the name of*
*			the primary index file for the target Pay*
*			file (no extension).
*			*
*	T_CT_NDX	GLOBAL	String variable used to store the name of*
*			the primary index file for the target
*			Enrollment totals file (no extension).
*			*

*			*
*	T_M_NDX_F	GLOBAL	String variable used to store the name of*
*			the primary index file for the target *
*			Master file (with extension). *
*			*
*	T_P_NDX_F	GLOBAL	String variable used to store the name of*
*			the primary index file for the target Pay*
*			file (with extension). *
*			*
*	T_CT_NDX_F	GLOBAL	String variable used to store the name of*
*			the primary index file for the target *
*			Enrollment totals file (with extension). *
*			*
*	LINE_NUM	GLOBAL	Variable used to keep track of the number*
*			of pay records that have been displayed *
*			on the screen. *
*			*
*	DISP_LINE	GLOBAL	Variable used to hold the value which *
*			corresponds to the specific line on the *
*			screen where the data will be displayed. *
*			*
*	SAV_REC1 -	GLOBAL	Used to save the database record numbers *
*	SAV_REC16		of the Pay records associated with the *
*			selected Master record. *
*			*
*	FLAG_REC1 -	GLOBAL	Boolean variables used to indicate which *
*	FLAG_REC16		associated Pay records the user has *
*			marked for deletion. *
*			*
*	ED_REC_NUM	GLOBAL	Used to save the database record number *
*			of the Pay record the user has selected *
*			for editing. *
*			*
*	LOW_DATE	GLOBAL	Used to save the ending date of the pay *
*			period for the Pay record which precedes *
*			the Pay record currently being processed. *
*			*
*	HIGH_DATE	GLOBAL	Used to save the beginning date of the *
*			pay period for the Pay record which fol- *
*			lows the Pay record currently being pro- *
*			cessed. *
*			*
*	GOOD_RO	GLOBAL	Boolean variable used to indicate whether*
*			all input relational operators are valid.*
*			*
*	BAD_SSAN	GLOBAL	Boolean variable used to indicate whether*
*			the input primary key is valid. *
*			*
*	S2-S7,S17,	GLOBAL	Used as spacing variables in the print *
*	S26,S31		format string variables. *
*			*
*	DCR_VAL	GLOBAL	Used to store the value multiplied by the*
*			DC_RATING (database variable) in deter- *
*			mining the WPSS score. *
*			*

*	GPA_VAL	GLOBAL	Used to store the value multiplied by the*
*			CUM_GPA (database variable) in deter- *
*			mining the WPSS score. *
*			
*	SAT_VAL	GLOBAL	Used to store the value multiplied by the*
*			SAT_CUM (database variable) in deter- *
*			mining the WPSS score. *
*			
*	AA_VAL	GLOBAL	Used to store the value multiplied by the*
*			AFOQT_AA (database variable) in deter- *
*			mining the WPSS score. *
*			
*	QUAN_VAL	GLOBAL	Used to store the value multiplied by the*
*			AFOQT_QUAN (database variable) in deter- *
*			mining the WPSS score. *
*			
*	VERB_VAL	GLOBAL	Used to store the value multiplied by the*
*			AFOQT_VERB (database variable) in deter- *
*			mining the WPSS score. *
*			
*	LOOP_CNTRL	LOCAL	Used to control exit from the main pro- *
*			gram loop. While TRUE, control remains *
*			within the loop. The variable is set to *
*			FALSE by either selecting options to re- *
*			turn to dBASE III or to return to DOS. *
*			
-----*			

```

PUBLIC F_PARA
PUBLIC G_PARA
PUBLIC R_PARA
PUBLIC QO_PARA
PUBLIC QS_PARA
PUBLIC F_SELECT
PUBLIC G_SELECT
PUBLIC R_SELECT
PUBLIC QO_SELECT
PUBLIC QS_SELECT
PUBLIC QUIT_KEY
PUBLIC M_CHOICE
PUBLIC P_CHOICE
PUBLIC VP_CHOICE
PUBLIC TQ_CHOICE
PUBLIC FILT_STR
PUBLIC T_FOR_STR
PUBLIC EMPTY_M
PUBLIC EMPTY_P
PUBLIC REC_NUM
PUBLIC DEL_FLAG
PUBLIC FIRST_TIME
PUBLIC IN_SSAN
PUBLIC IN_FNAM
PUBLIC IN_MNAM
PUBLIC IN_LNAM

```

PUBLIC T\_PHON  
PUBLIC T\_PATH  
PUBLIC M\_FILE  
PUBLIC P\_FILE  
PUBLIC CT\_FILE  
PUBLIC NDX\_LIST  
PUBLIC M\_NDX  
PUBLIC P\_NDX  
PUBLIC CT\_NDX  
PUBLIC M\_NDX\_F  
PUBLIC P\_NDX\_F  
PUBLIC CT\_NDX\_F  
PUBLIC M\_NDX\_STR  
PUBLIC P\_NDX\_STR  
PUBLIC M\_FORM\_STR  
PUBLIC DEST\_FILE  
PUBLIC T\_M\_FILE  
PUBLIC T\_P\_FILE  
PUBLIC T\_CT\_FILE  
PUBLIC T\_M\_NDX  
PUBLIC T\_P\_NDX  
PUBLIC T\_CT\_NDX  
PUBLIC T\_M\_NDX\_F  
PUBLIC T\_P\_NDX\_F  
PUBLIC T\_CT\_NDX\_F  
PUBLIC LINE\_NUM  
PUBLIC DISP\_LINE  
PUBLIC SAV\_REC1  
PUBLIC SAV\_REC2  
PUBLIC SAV\_REC3  
PUBLIC SAV\_REC4  
PUBLIC SAV\_REC5  
PUBLIC SAV\_REC6  
PUBLIC SAV\_REC7  
PUBLIC SAV\_REC8  
PUBLIC SAV\_REC9  
PUBLIC SAV\_REC10  
PUBLIC SAV\_REC11  
PUBLIC SAV\_REC12  
PUBLIC SAV\_REC13  
PUBLIC SAV\_REC14  
PUBLIC SAV\_REC15  
PUBLIC SAV\_REC16  
PUBLIC FLAG\_REC1  
PUBLIC FLAG\_REC2  
PUBLIC FLAG\_REC3  
PUBLIC FLAG\_REC4  
PUBLIC FLAG\_REC5  
PUBLIC FLAG\_REC6  
PUBLIC FLAG\_REC7  
PUBLIC FLAG\_REC8  
PUBLIC FLAG\_REC9  
PUBLIC FLAG\_REC10  
PUBLIC FLAG\_REC11  
PUBLIC FLAG\_REC12

```

PUBLIC FLAG_REC13
PUBLIC FLAG_REC14
PUBLIC FLAG_REC15
PUBLIC FLAG_REC16
PUBLIC ED_REC_NUM
PUBLIC LOW_DATE
PUBLIC HIGH_DATE
PUBLIC GOOD_RO
PUBLIC BAD_SSAN
PUBLIC S2
PUBLIC S3
PUBLIC S4
PUBLIC S5
PUBLIC S6
PUBLIC S7
PUBLIC S17
PUBLIC S26
PUBLIC S31
PUBLIC DCR_VAL
PUBLIC GPA_VAL
PUBLIC SAT_VAL
PUBLIC AA_VAL
PUBLIC QUAN_VAL
PUBLIC VERB_VAL
PRIVATE LOOP_CNTRL

```

```

* Start program code.  *

```

```

* Set dBASE III PLUS status line off.  *

```

```

SET STATUS OFF

```

```

* Set dBASE III PLUS bottom line off.  *

```

```

SET SCOREBOARD OFF

```

```

* Display initial screen.  *

```

```

@ 1, 0 TO 3,79
@ 2,22 SAY 'ROTC CADET INFORMATION SYSTEM (RCIS)'
@ 4, 0 TO 18,79
@ 6,33 SAY 'Version 1.10'
@ 8,38 SAY 'by'
@ 10,31 SAY 'Carter L. Frank'
@ 12,27 SAY 'The University of Arizona'
@ 14,18 SAY 'Department of Management Information Systems'
@ 16,31 SAY 'Copyright (C) 1987'
@ 20,29 TO 22,50 DOUBLE

```

```

* Set video attributes to blink.  *

```

```

SET COLOR TO W*/N

```

```

@ 21,30 SAY ' INITIALIZING  RCIS '
@ 24,0

```

```

* Initialize RCIS. *

* Designate RCIS_P1.PRG as active procedure file. *

SET PROCEDURE TO RCIS_P1

* Call procedure INIT from RCIS_P1.PRG *

DO INIT
A_SELECT = ''
LOOP_CNTRL = .T.

* Restore default video attributes. *

SET COLOR TO
@ 4, 0 CLEAR TO 24,79
PROC_VAL = 0

* Main Program Loop for RCIS. *

DO WHILE (LOOP_CNTRL)

* If the function sequence code is not "escape", reset sequence code *
* to start and reset function selected code to "add". *

IF (SUBSTR(F_PARA,1,1) <> 'C')
    F_PARA = STUFF(F_PARA,1,1,'A')
    F_PARA = STUFF(F_PARA,6,1,'H')
ENDIF

* While a function has not been selected, do the following. *

DO WHILE (SUBSTR(F_PARA,1,1) <> 'B')

    * Clear menus to the right of the function menu. *

    @ 4,19 CLEAR TO 24,79

    * Clear the text display area. *

    @ 18, 0 CLEAR TO 24,79

    * Display "Select Function" box. *

    @ 20, 1 TO 22,17
    @ 21, 2 SAY 'SELECT FUNCTION'

    * Call menu assembly routine, passing function menu parameter. *

    CALL MENU WITH F_PARA
    @ 24, 0

    * Get function choice from returned parameter. *

    F_SELECT = SUBSTR(F_PARA,6,1)

```

DO CASE

\* If function selected is not "Return to dBASE" or "Exit to DOS" \*  
\* continue with the following. \*

CASE F\_SELECT <= 'M'

\* Initialize group menu sequence code and starting position.\*

G\_PARA = STUFF(G\_PARA,1,1,'A')  
G\_PARA = STUFF(G\_PARA,6,1,'H')

\* While a group has not been selected, do the following: \*

DO WHILE SUBSTR(G\_PARA,1,1) <> 'B'

\* Clear text display area and display "Select Group" box.\*

@ 18, 0 CLEAR TO 24,79  
@ 20,19 TO 22,32  
@ 21,20 SAY 'SELECT GROUP'

\* Call menu assembly routine, passing group parameter. \*

CALL MENU WITH G\_PARA  
@ 24,0

\* Get group selected code. \*

G\_SELECT = SUBSTR(G\_PARA,6,1)

\* If no group selected, then "escape" sequence has been \*  
\* pressed. Set function sequence code to "escape" and \*  
\* exit this loop. Control returns to select function \*  
\* loop above. \*

IF SUBSTR(G\_PARA,1,1) = 'A'  
F\_PARA = STUFF(F\_PARA,1,1,'C')  
EXIT  
ENDIF  
QS\_SELECT = ''

\* If function select is Query, continue with the \*  
\* following: \*

IF (F\_SELECT = 'M')

\* Initialize query select menu sequence code and \*  
\* starting position. \*

QS\_PARA = STUFF(QS\_PARA,1,1,'A')  
QS\_PARA = STUFF(QS\_PARA,6,1,'H')

\* While a query type has not been selected, do the \*  
\* following: \*



```

DO WHILE SUBSTR(QS_PARA,1,1) <> 'B'

    * Clear text display area and display *
    * "Select Query" box. *

    @ 18, 0 CLEAR TO 24, 79
    @ 20,38 TO 22,51
    @ 21,39 SAY 'SELECT QUERY'

    * Call menu assembly routine, passing query select *
    * parameter. *

    CALL MENU WITH QS_PARA
    @ 24,0

    * Get query selected code. *

    QS_SELECT = SUBSTR(QS_PARA,6,1)

    * If no query selected, then "escape" sequence has *
    * been pressed. Set function sequence code to *
    * "escape" and exit this loop. Control returns to *
    * select group loop above. *

    IF SUBSTR(QS_PARA,1,1) = 'A'
        G_PARA = STUFF(G_PARA,1,1,'C')
        EXIT
    ENDIF

    * Initialize query output menu sequence code and *
    * starting position. *

    QO_PARA = STUFF(QO_PARA,1,1,'A')
    QO_PARA = STUFF(QO_PARA,6,1,'H')

    * While a query output has not been selected, do *
    * the following: *

    DO WHILE SUBSTR(QO_PARA,1,1) <> 'B'

        * Clear text display area and display *
        * "Select Output Media" box. *

        @ 18, 0 CLEAR TO 24, 79
        @ 20,56 TO 22,76
        @ 21,57 SAY 'SELECT OUTPUT MEDIA'

        * Call menu assembly routine, passing query *
        * output parameter. *

        CALL MENU WITH QO_PARA
        @ 24,0

        * Get query output media code. *

```

```

        QO_SELECT = SUBSTR(QO_PARA,6,1)

        * If no query output selected, then "escape" *
        * sequence has been pressed. Set function *
        * sequence code to "escape" and exit this loop.*
        * Control returns to select query loop above. *

        IF SUBSTR(QO_PARA,1,1) = 'A'
            QS_PARA = STUFF(QS_PARA,1,1,'C')
            EXIT
        ENDIF
    ENDDO
ENDDO

* If function select is not Query and not Transfer, *
* continue with the following: *

ELSE
    IF (F_SELECT <> 'L')

        * Initialize record menu sequence code and *
        * starting position. *

        R_PARA = STUFF(R_PARA,1,1,'A')
        R_PARA = STUFF(R_PARA,6,1,'H')

        * While a record has not been selected, do *
        * the following: *

        DO WHILE SUBSTR(R_PARA,1,1) <> 'B'

            * Clear text display area and display *
            * "Select Record" box. *

            @ 18, 0 CLEAR TO 24,79
            @ 20,36 TO 22,50
            @ 21,37 SAY 'SELECT RECORD'

            * Call menu assembly routine, passing query *
            * output parameter. *

            CALL MENU WITH R_PARA
            @ 24,0

            * Get record code. *

            R_SELECT = SUBSTR(R_PARA,6,1)

            * If no record selected, then "escape" sequence *
            * has been pressed. Set function sequence code *
            * to "escape" and exit this loop. Control *
            * returns to select group loop above. *

            IF SUBSTR(R_PARA,1,1) = 'A'

```

```

        G_PARA = STUFF(G_PARA,1,1,'C')
        EXIT
    ENDIF
ENDDO
ENDIF
ENDIF
ENDDO

* If a function has been selected, then transfer control *
* to the appropriate procedure file. *

IF SUBSTR(F_PARA,1,1) = 'B'

    * If the function selected was either "Add" or "Edit", *
    * then pull in the WPSS multiplier values to be used by *
    * those functions. *

    IF (F_SELECT = 'H') .OR. (F_SELECT = 'I')
        SELECT 1
        USE T_CDT_WP
        GO TOP
        DCR_VAL = DCR_NUM
        GPA_VAL = GPA_NUM
        SAT_VAL = SAT_NUM
        AA_VAL = AA_NUM
        QUAN_VAL = QUAN_NUM
        VERB_VAL = VERB_NUM
        SELECT 1
        USE
    ENDIF

    * If the function selected was previous to "Query" and *
    * and RCIS_P2.PRG is not the active procedure file, *
    * designate RCIS_P2.PRG as active and clear the bottom *
    * of the screen. *

    IF ((F_SELECT <= 'L') .AND. (PROC_VAL <> 2))
        SET PROCEDURE TO RCIS_P2
        PROC_VAL = 2
        @ 18, 0 CLEAR TO 24,79
        @ 21,33 SAY 'OPENING FILES'
        @ 24, 0
    ENDIF
DO CASE
CASE F_SELECT = 'H'
    DO ADD_REC
CASE F_SELECT = 'I'
    DO EDIT_REC
CASE F_SELECT = 'J'
    DO VIEW_REC
CASE F_SELECT = 'K'
    DO DEL_REC
CASE F_SELECT = 'L'
    DO TRANS_REC
CASE F_SELECT = 'M'

```

```

@ 18, 0 CLEAR TO 24,79
@ 23,18 SAY 'BUILDING QUERY INPUT MENU. PLEASE WAIT.'
@ 24, 0

* If the function selected was "Query" and *
* RCIS_P3.PRG is not the active procedure file, *
* designate RCIS_P3.PRG as active and call query *
* main driver procedure. *

IF (PROC_VAL <> 3)
    SET PROCEDURE TO RCIS_P3
    PROC_VAL = 3
ENDIF
DO QUERIES

ENDCASE
ENDIF

* If either "Exit to dBASE" or "Exit to DOS" was selected, then *
* exit the main control loop. *

CASE (F_SELECT = 'N') .OR. (F_SELECT = 'O')
    LOOP_CNTRL = .F.
    EXIT
ENDCASE
ENDDO
ENDDO

* Decouple MENU.BIN from the program. *

RELEASE MODULE MENU

* If "Exit to dBASE" was selected, restore initial dBASE environment. *
* Otherwise return to DOS. *

IF F_SELECT = 'N'
    SET CONFIRM OFF
    SET SCOREBOARD ON
    SET TALK ON
    SET ESCAPE ON
    SET SAFETY ON
    SET BELL ON
    SET STATUS ON
    CLEAR ALL
ELSE
    CLEAR ALL
    QUIT
ENDIF

* End of Main Program. *

RETURN

```

```

*-----*
*               BEGINNING OF RCIS_P1.PRG               *
*-----*
*               INIT                                    *
*-----*
* SUMMARY:                                             *
*     INIT is the main initialization procedure for RCIS. It calls *
*     routines that initialize variables accessed by the RCIS main *
*     program.                                         *
*-----*
* CALLED PROCEDURES:                                  *
*
*               Procedure Name                        Location      *
*               -----
*               SET_MENU                             RCIS_P1.PRG      *
*               BOX_CHAR                             RCIS_P1.PRG      *
*               F_MENU                               RCIS_P1.PRG      *
*               G_MENU                               RCIS_P1.PRG      *
*               R_MENU                               RCIS_P1.PRG      *
*               QS_MENU                             RCIS_P1.PRG      *
*               QO_MENU                             RCIS_P1.PRG      *
*-----*

```

#### PROCEDURE INIT

```

*
DO SET_MENU
DO BOX_CHAR
DO F_MENU
DO G_MENU
DO R_MENU
DO QS_MENU
DO QO_MENU
*
RETURN

```

```

*-----*
*                               SET_MENU                               *
*-----*
*
* SUMMARY:
*   The SET_MENU procedure establishes the application program
*   environment.  The environment includes the following features:
*
*       1. Deleted records are not displayed.
*       2. The user must press enter to "confirm" input is complete.
*       3. Date variables do not display the century.
*       4. The system bell is turned off.
*       5. Interactive system messages are turned off.
*       6. Files will be overwritten without system warning prompts.
*       7. The assembly routine, MENU.BIN, is coupled to the program as
*          a callable subroutine.
*
*-----*

```

# PROCEDURE SET\_MENU

```

*
SET DELETED OFF
SET CONFIRM ON
SET CENTURY OFF
SET BELL OFF
SET TALK OFF
SET ESCAPE OFF
SET SAFETY OFF
SET DATE BRITISH
LOAD MENU
*
RETURN

```

```

*-----*
*                                BOX_CHAR                                *
*-----*
*
* SUMMARY:
*      The BOX_CHAR procedure initializes variables that define the
*      special graphics characters used to create the menu boxes
*      imbedded in the parameter string passed to MENU.BIN.
*
* VARIABLE DECLARATIONS:
*
*      Variable Name      Status      Purpose
*      -----
*      TL_BOX            GLOBAL      Defines top left corner of menu box.
*
*      TR_BOX            GLOBAL      Defines top right corner of menu box.
*
*      BL_BOX            GLOBAL      Defines bottom left corner of menu box.
*
*      BR_BOX            GLOBAL      Defines bottom right corner of menu box.
*
*      LM_BOX            GLOBAL      Defines left T-bar used to separate the
*                                   menu title from the menu body.
*
*      RM_BOX            GLOBAL      Defines right T-bar used to separate the
*                                   menu title from the menu body.
*
*      V_BAR             GLOBAL      Defines a vertical bar.
*
*      X_BAR             GLOBAL      Defines 5 character double horizontal bar
*
*      X_BAR1            GLOBAL      Defines 10 character double horizontal bar
*
*      X_BAR2            GLOBAL      Defines 12 character double horizontal bar
*
*      X_BAR3            GLOBAL      Defines 14 character double horizontal bar
*
*      X_BAR4            GLOBAL      Defines 15 character double horizontal bar
*
*      X_BAR5            GLOBAL      Defines 17 character double horizontal bar
*-----*

```

# PROCEDURE BOX\_CHAR

```

*
PUBLIC TL_BOX
PUBLIC TR_BOX
PUBLIC BL_BOX
PUBLIC BR_BOX
PUBLIC LM_BOX
PUBLIC RM_BOX
PUBLIC V_BAR
PUBLIC X_BAR
PUBLIC X_BAR1

```

```

PUBLIC X_BAR2
PUBLIC X_BAR3
PUBLIC X_BAR4
PUBLIC X_BAR5
*
* ASSIGN SPECIAL GRAPHICS CHARACTERS
*
TL_BOX  = CHR(201)
TR_BOX  = CHR(187)
BL_BOX  = CHR(200)
BR_BOX  = CHR(188)
LM_BOX  = CHR(204)
RM_BOX  = CHR(185)
V_BAR   = CHR(186)
X_BAR   = CHR(205) + CHR(205) + CHR(205) + CHR(205) + CHR(205)
X_BAR1  = X_BAR + X_BAR
X_BAR2  = X_BAR1 + CHR(205) + CHR(205)
X_BAR3  = X_BAR2 + CHR(205) + CHR(205)
X_BAR4  = X_BAR3 + CHR(205)
X_BAR5  = X_BAR4 + CHR(205) + CHR(205)
*
RETURN

```



```

*-----*
*                                     F_MENU                                     *
*-----*
*
* SUMMARY:
*
*   The F_MENU procedure initializes the string parameter, F_PARA,
*   that is passed to MENU.BIN to create the function menu. The
*   string parameter consists of two parts. The first seven charac-
*   ters constitute a header that provides control information for the
*   assembly routine. These control functions are discussed in detail
*   within the VARIABLE DECLARATION section that follows. The remain-
*   ing characters (up to 237) constitute text data that represents
*   the actual menu box that will be displayed by the assembly menu
*   driver routine.
*
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   SEQ_1              LOCAL      The first character of the header is the
*                                   sequence code. The menu driver responds
*                                   to the following codes:
*
*                                   A = Initial sequence. Paint the menu and
*                                   accept user input. If this code is
*                                   returned from MENU.BIN, it means the
*                                   user pressed the <Esc> key to abort
*                                   menu selection. In this event, a
*                                   "roll back" to the previous menu is
*                                   initiated.
*
*                                   B = This code is returned when a menu
*                                   selection was made by the user. If
*                                   this code is sent to MENU.BIN, the
*                                   menu box is repainted and an early
*                                   exit is made without accepting user
*                                   input.
*
*                                   C = This code is sent to MENU.BIN to
*                                   signal a "roll back" to a previous
*                                   menu. The menu driver will erase
*                                   menu frames to the right of the
*                                   current menu, and new user input is
*                                   accepted.
*
*   ACT_1              LOCAL      The second character in the header is the
*                                   active menu flag. It is used by MENU.BIN
*                                   to determine whether "roll back" will be
*                                   recognized by pressing the <Esc> key.
*                                   The only menu that does not permit
*                                   "roll back" is the function menu. Setting
*                                   this flag to A indicates the "roll back"
*                                   is disabled.
*
*   SROW_1             LOCAL      The third character in the header is the

```

```

*          row to start the menu box. The value is *
*          computed relative to A = 0. *
*          *
*          SCOL_1          LOCAL          The fourth character in the header is the*
*          *          *          column to start the menu box. Its value *
*          *          *          is also computed relative to A = 0. *
*          *          *          *
*          BROW_1          LOCAL          The fifth character in the header is the *
*          *          *          bottom row of the menu box. Its value is*
*          *          *          also computed relative to A = 0. *
*          *          *          *
*          AROW_1          LOCAL          The sixth character in the header is the *
*          *          *          row that was active when the user either *
*          *          *          pressed the <Enter> key for selecting a *
*          *          *          function or pressed the <Esc> key to *
*          *          *          abort the current menu. By inspection *
*          *          *          this position, the program can determine *
*          *          *          the menu item that the user selected. *
*          *          *          *
*          SLEN_1          LOCAL          The seventh character in the header is *
*          *          *          the menu field width(or character length)*
*          *          *          Total width includes the two graphic box *
*          *          *          characters. The value is also computed *
*          *          *          relative to A = 0. *
*          *          *          *
*-----*

```

#### PROCEDURE F\_MENU

```

*
* ASSIGN FUNCTION MENU PARAMETER
*
PRIVATE SEQ_1
PRIVATE ACT_1
PRIVATE SROW_1
PRIVATE SCOL_1
PRIVATE BROW_1
PRIVATE AROW_1
PRIVATE SLEN_1
*
SEQ_1  = CHR(65 + 0)
ACT_1  = CHR(64 + 1)
SROW_1 = CHR(65 + 4)
SCOL_1 = CHR(65 + 4)
BROW_1 = CHR(65 + 15)
AROW_1 = CHR(65 + 7)
SLEN_1 = CHR(65 + 12)
*
F_PARA = SEQ_1 + ACT_1 + SROW_1 + SCOL_1 + BROW_1 + AROW_1 + SLEN_1
F_PARA = F_PARA + TL_BOX + X_BAR1 + TR_BOX
F_PARA = F_PARA + V_BAR + ' FUNCTION ' + V_BAR
F_PARA = F_PARA + LM_BOX + X_BAR1 + RM_BOX
F_PARA = F_PARA + V_BAR + ' Add ' + V_BAR
F_PARA = F_PARA + V_BAR + ' Edit ' + V_BAR
F_PARA = F_PARA + V_BAR + ' View ' + V_BAR
F_PARA = F_PARA + V_BAR + ' Delete ' + V_BAR

```

```
F_PARA = F_PARA + V_BAR + ' Transfer ' + V_BAR
F_PARA = F_PARA + V_BAR + ' Query ' + V_BAR
F_PARA = F_PARA + V_BAR + ' dBASE ' + V_BAR
F_PARA = F_PARA + V_BAR + ' Exit ' + V_BAR
F_PARA = F_PARA + BL_BOX + X_BAR1 + BR_BOX
*
RETURN
```

```

*-----*
*                                     G_MENU                                     *
*-----*
*
* SUMMARY:
*      The G_MENU procedure initializes the string parameter, G_PARA,
*      that is passed to MENU.BIN to create the group menu. The string
*      parameter construction is identical to that specified in F_MENU
*      for the function menu.
*
*-----*

```

# PROCEDURE G\_MENU

```

*
* ASSIGN GROUP MENU PARAMETERS
*
PRIVATE SEQ_1
PRIVATE ACT_1
PRIVATE SROW_1
PRIVATE SCOL_1
PRIVATE BROW_1
PRIVATE AROW_1
PRIVATE SLEN_1
*
SEQ_1  = CHR(65 + 0)
ACT_1  = CHR(64 + 2)
SROW_1 = CHR(65 + 4)
SCOL_1 = CHR(65 + 20)
BROW_1 = CHR(65 + 9)
AROW_1 = CHR(65 + 7)
SLEN_1 = CHR(65 + 12)
*
G_PARA = SEQ_1 + ACT_1 + SROW_1 + SCOL_1 + BROW_1 + AROW_1 + SLEN_1
G_PARA = G_PARA + TL_BOX + X_BAR1 + TR_BOX
G_PARA = G_PARA + V_BAR + ' GROUPS ' + V_BAR
G_PARA = G_PARA + LM_BOX + X_BAR1 + RM_BOX
G_PARA = G_PARA + V_BAR + ' Active ' + V_BAR
G_PARA = G_PARA + V_BAR + ' Inactive ' + V_BAR
G_PARA = G_PARA + BL_BOX + X_BAR1 + BR_BOX
*
RETURN

```

```

*-----*
*                                     R_MENU                                     *
*-----*
*
* SUMMARY:                                                                    *
* The R_MENU procedure initializes the string parameter, R_PARA,             *
* that is passed to MENU.BIN to create the group menu. The string           *
* parameter construction is identical to that specified in F_MENU            *
* for the function menu.                                                     *
*-----*

```

# PROCEDURE R\_MENU

```

*
* ASSIGN RECORD MENU PARAMETERS
*
PRIVATE SEQ_1
PRIVATE ACT_1
PRIVATE SROW_1
PRIVATE SCOL_1
PRIVATE BROW_1
PRIVATE AROW_1
PRIVATE SLEN_1
*
SEQ_1  = CHR(65 + 0)
ACT_1  = CHR(64 + 3)
SROW_1 = CHR(65 + 4)
SCOL_1 = CHR(65 + 36)
BROW_1 = CHR(65 + 9)
AROW_1 = CHR(65 + 7)
SLEN_1 = CHR(65 + 16)
*
R_PARA = SEQ_1 + ACT_1 + SROW_1 + SCOL_1 + BROW_1 + AROW_1 + SLEN_1
R_PARA = R_PARA + TL_BOX + X_BAR3 + TR_BOX
R_PARA = R_PARA + V_BAR + '  RECORDS  ' + V_BAR
R_PARA = R_PARA + LM_BOX + X_BAR3 + RM_BOX
R_PARA = R_PARA + V_BAR + ' Cadet Master ' + V_BAR
R_PARA = R_PARA + V_BAR + ' Cadet Pay   ' + V_BAR
R_PARA = R_PARA + BL_BOX + X_BAR3 + BR_BOX
*
RETURN

```

```

*-----*
*                                     QS_MENU                                     *
*-----*
*
* SUMMARY:
*       The QS_MENU procedure initializes the string parameter, QS_PARA,
*       that is passed to MENU.BIN to create the group menu. The string
*       parameter construction is identical to that specified in F_MENU
*       for the function menu.
*-----*

```

# PROCEDURE QS\_MENU

```

*
* ASSIGN QUERY SELECTION MENU PARAMETERS
*
PRIVATE SEQ_1
PRIVATE ACT_1
PRIVATE SROW_1
PRIVATE SCOL_1
PRIVATE BROW_1
PRIVATE AROW_1
PRIVATE SLEN_1
*
SEQ_1  = CHR(65 + 0)
ACT_1  = CHR(64 + 4)
SROW_1 = CHR(65 + 4)
SCOL_1 = CHR(65 + 36)
BROW_1 = CHR(65 + 17)
AROW_1 = CHR(65 + 7)
SLEN_1 = CHR(65 + 17)
*
QS_PARA = SEQ_1 + ACT_1 + SROW_1 + SCOL_1 + BROW_1 + AROW_1 + SLEN_1
QS_PARA = QS_PARA + TL_BOX + X_BAR4 + TR_BOX
QS_PARA = QS_PARA + V_BAR + '  QUERY  TYPE  ' + V_BAR
QS_PARA = QS_PARA + LM_BOX + X_BAR4 + RM_BOX
QS_PARA = QS_PARA + V_BAR + '  WPSS Info  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  Schlrsbp Qual  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  DOC Fiscal Yr  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  AS Class Info  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  2-Yr Pgm Cand  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  Com Date Susp  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  Schlrsbp Expr  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  Weigh/Aerobic  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  Individual  ' + V_BAR
QS_PARA = QS_PARA + V_BAR + '  Pay Info  ' + V_BAR
QS_PARA = QS_PARA + BL_BOX + X_BAR4 + BR_BOX
*
RETURN

```

```

*-----*
*                               QO_MENU                               *
*-----*
*
* SUMMARY:
*       The QO_MENU procedure initializes the string parameter, QO_PARA,
*       that is passed to MENU.BIN to create the group menu. The string
*       parameter construction is identical to that specified in F_MENU
*       for the function menu.
*
*-----*

```

# PROCEDURE QO\_MENU

```

*
* ASSIGN QUERY OUTPUT MENU PARAMETERS
*
PRIVATE SEQ_1
PRIVATE ACT_1
PRIVATE SROW_1
PRIVATE SCOL_1
PRIVATE BROW_1
PRIVATE AROW_1
PRIVATE SLEN_1
*
SEQ_1  = CHR(65 + 0)
ACT_1  = CHR(64 + 3)
SROW_1 = CHR(65 + 4)
SCOL_1 = CHR(65 + 57)
BROW_1 = CHR(65 + 10)
AROW_1 = CHR(65 + 7)
SLEN_1 = CHR(65 + 19)
*
QO_PARA = SEQ_1 + ACT_1 + SROW_1 + SCOL_1 + BROW_1 + AROW_1 + SLEN_1
QO_PARA = QO_PARA + TL_BOX + X_BAR5 + TR_BOX
QO_PARA = QO_PARA + V_BAR + ' QUERY OUTPUT ' + V_BAR
QO_PARA = QO_PARA + LM_BOX + X_BAR5 + RM_BOX
QO_PARA = QO_PARA + V_BAR + ' 80-Col Screen ' + V_BAR
QO_PARA = QO_PARA + V_BAR + ' 80-Col Printer ' + V_BAR
QO_PARA = QO_PARA + V_BAR + ' 132-Col Printer ' + V_BAR
QO_PARA = QO_PARA + BL_BOX + X_BAR5 + BR_BOX
*
RETURN

```





```

IF (QUIT_KEY)
    EXIT
ENDIF
DO INIT_DB
SELECT 1
IF (EMPTY_M)
    DO CASE

        * If the Master file is empty and the user has selected a *
        * Master record for processing, build the index list and *
        * continue processing the users database request. *

        CASE R_SELECT = 'H'
            IF (.NOT. FILE(M_NDX_F))
                INDEX ON &M_NDX_STR TO &M_NDX
            ENDIF
            DO BLD_NDX WITH M_NDX
            SET INDEX TO &NDX_LIST

            * If the Master file is empty and the user has selected a *
            * Pay record for processing, automatically exit this func- *
            * tion and return to the select function menu. *

            CASE R_SELECT = 'I'
                @ 22, 0
                ? CHR(7)
                @ 23, 4 SAY 'MASTER FILE IS EMPTY.  PRESS ANY KEY TO CONTINUE.'
                WAIT ''
                EXIT

            ENDCASE
        ENDIF
        SET FILTER TO &FILT_STR

        * Issue dBASE III PLUS command to go to the record which matches *
        * the primary key value. *

        SEEK IN_SSAN
        DO CASE

            * If a matching Master record is found, set up the screen format *
            * and prepare all files required to process the record display. *

            CASE EOF()
                DO CASE

                    * If the user has selected to process a Master record, *
                    * issue the dBASE III PLUS commands that coordinate the *
                    * interaction between the supporting files and the main *
                    * file. *

                    CASE R_SELECT = 'H'
                        @ 22, 0
                        @ 23, 0
                        @ 23,20 SAY 'PREPARING DATABASE FILE FOR NEW RECORD.'
                        SET FORMAT TO &M_FORM_STR

```

```

APPEND BLANK
REC_NUM = RECNO()
REPLACE SSAN WITH IN_SSAN
SET SCOREBOARD ON
SET ESCAPE OFF
SET CONFIRM OFF
CLAS_NUM = ' ? '

```

\* Issue 'CHANGE' command to display the record data. \*

```

CHANGE
SET CONFIRM ON
GOTO REC_NUM
IF PERM_STRT = 'SAME'
    REPLACE PERM_STRT WITH LOCAL_STRT
    REPLACE PERM_CITY WITH LOCAL_CITY
    REPLACE PERM_STAT WITH 'AZ'
    REPLACE PERM_ZIP WITH LOCAL_ZIP
    T_PHON = '602' + LOCAL_PHON
    REPLACE PERM_PHON WITH T_PHON
ENDIF
REPLACE WPSS WITH ((DC_RTNG*DCR_VAL)+(CUM_GPA*100.00*GPA_VAL);
+ (SAT_CUM*SAT_VAL)+(AFOQT_AA*AA_VAL)+(AFOQT_QUAN*QUAN_VAL);
+ (AFOQT_VERB*VERB_VAL))
DO HGHT_CHK
IN_FNAM = F_NAME
IN_MNAM = M_NAME
IN_LNAM = L_NAME
DO RCIS_HDR

```

```

* If Master record was deleted by pressing the *
* <Ctrl> <U> keys, then prompt the user to see *
* if they really want to delete the record. If *
* they do, delete it; if not, recall is back to *
* current status. *

```

```

IF DELETED()
DO D_PROMPT
IF P_CHOICE
    @ 23, 0
    @ 23,23 SAY 'DELETING MASTER RECORD'
    PACK
    DEL_FLAG = .T.
ELSE
    RECALL RECORD REC_NUM
    P_CHOICE = .T.
    GOTO REC_NUM
ENDIF
ENDIF
IF (.NOT. DEL_FLAG)
CLAS_VAL = AS_CLASS
SET FILTER TO
COUNT FOR AS_CLASS = CLAS_VAL TO CLAS_TOT

SELECT 3

```

\*

```

        SEEK CLAS_VAL
        IF (.NOT. EOF())
            REPLACE AS_CL_TOT WITH CLAS_TOT
        ENDIF
    ENDIF
    IF (.NOT. DELETED()) .AND. (.NOT. DEL_FLAG)
        DO P_PROMPT
        IF (P_CHOICE)
            DO ADD_PAY
        ENDIF
    ENDIF
ENDIF

```

```

* If a Master record was not found for the input primary *
* key and the user has selected a Pay record for pro- *
* cessing, prompt the user to either try again or to *
* exit this function. *

```

```

CASE R_SELECT = 'I'
    @ 22, 0
    @ 23, 4 SAY 'MASTER '
    DO ERR_NF
    IF (M_CHOICE)
        LOOP
    ELSE
        EXIT
    ENDIF

```

```

ENDCASE

```

```

*

```

```

CASE (.NOT. EOF())
DO CASE

```

```

* If a matching Master record is found and the user has *
* selected a Master record for processing, prompt the *
* user to either try again or to exit the function. *

```

```

CASE (R_SELECT = 'H')
    DO ERR_RE
    IF (M_CHOICE)
        LOOP
    ELSE
        EXIT
    ENDIF

```

```

* If a matching Master record is found and the user has *
* selected a Pay record for processing, invoke the ADD_PAY *
* procedure and continue processing the user's request. *

```

```

CASE (R_SELECT = 'I')
    IN_FNAM = F_NAME
    IN_MNAM = M_NAME
    IN_LNAM = L_NAME
    @ 22, 0
    @ 23, 0

```

```

    @ 23,20 SAY 'SEARCHING DATABASE FILE FOR EXISTING PAY RECORDS.'
    DO ADD_PAY

```

```

                ENDCASE
        ENDCASE

        *   Give the user the opportunity to execute this function again.  *

        DO M_PROMPT
    ENDDO

    *   Close the database files used in this function.  *

    SELECT 3
    USE
    SELECT 2
    USE
    SELECT 1
    USE
    CLOSE FORMAT
*
    F_PARA = STUFF(F_PARA,1,1,'C')
    @ 21, 0
    ON ERROR
*
    RETURN

```

```

*-----*
*                                ADD_PAY                                *
*-----*
*
* SUMMARY:
*
*   The ADD_PAY procedure adds new subordinate (Pay) records to rela-
*   tions within RCIS. This procedure is controlled by the ADD_REC
*   procedure and is only invoked after the controlling procedure has
*   determined that all required conditions have been met. This pro-
*   cedure edit checks the pay date periods to ensure they don't over-
*   lap and it allows the user to add up to 16 (maximum) Pay records
*   to any one Master record. This procedure is terminated when the
*   user enters a <N> in the ADD field displayed on the screen.
*
* CALLED PROCEDURES:
*
*                                Procedure Name      Location
*                                -----
*                                RCIS_HDR             RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   END_DATE           LOCAL       Used to save the ending pay date from the
*                                   previous pay period so it can be compared
*                                   to the current beginning date.
*
*   ADD_MORE           LOCAL       Boolean flag which indicates whether to
*                                   add the input pay record or to terminate
*                                   the add and the procedure.
*
*-----*

```

# ``` PROCEDURE ADD_PAY ```

```

*
  PRIVATE END_DATE
  PRIVATE ADD_MORE
*
  SELECT 2

  * If the Pay file is empty, set up the index file and *
  * continue processing the users database request.      *

  IF (EMPTY_P)
    IF (.NOT. FILE(P_NDX_F))
      INDEX ON &P_NDX_STR TO &P_NDX
    ENDIF
    SET INDEX TO &P_NDX
  ENDIF
  SET SCOREBOARD ON
  SET ESCAPE ON
  CLEAR TYPEAHEAD

  * Build the screen header for this function. *

```

```

@ 1, 0 TO 3,79 DOUBLE
@ 2,25 SAY 'INDIVIDUAL CADET DATA - PAY INFORMATION'
@ 2, 2 SAY TRIM(LEFT(IN_LNAM,10))+', ' +LEFT(IN_FNAM,1)+' ' +LEFT(IN_MNAM,1)
@ 4, 0 SAY '      REC   BEGINNING   ENDING           RESID    BOOK   ' ;
      + '      FT    ATP    FSP '
@ 5, 0 SAY 'ADD #    PAY DATE    PAY DATE    TUITION (I OR O) FEES   ' ;
      + '    DAYS    DAYS    DAYS'
DISP_LINE = 1
LINE_NUM = 6
END_DATE = CTOD('01/01/01')
SET FILTER TO &FILT_STR

```

```

* Issue dBASE III PLUS command to go to the record which matches *
* the primary key value.                                         *

```

```

SEEK IN_SSAN
IF (.NOT. EOF())

```

```

* Display the associated Pay records that already exist. *

```

```

DO WHILE ((.NOT. EOF()) .AND. (LINE_NUM <= 22))
@ LINE_NUM, 5 SAY LTRIM(STR(DISP_LINE))
@ LINE_NUM,10 SAY PAY_DATE1
@ LINE_NUM,22 SAY PAY_DATE2
@ LINE_NUM,33 SAY TUITION
@ LINE_NUM,45 SAY RES_STATUS
@ LINE_NUM,52 SAY BOOK_FEES
@ LINE_NUM,62 SAY FT_DAYS
@ LINE_NUM,69 SAY ATP_DAYS
@ LINE_NUM,76 SAY FSP_DAYS
DISP_LINE = DISP_LINE + 1
LINE_NUM = LINE_NUM + 1
END_DATE = PAY_DATE2

```

```

* Go to the next database record which matches the primary key. *

```

```

SKIP
ENDDO
ENDIF
ADD_MORE = .T.
IF (LINE_NUM > 22)
? CHR(7)
@ 23, 0 SAY 'MAX # OF PAY RECORDS HAVE BEEN ADDED. PRESS ANY KEY TO' ;
      + ' CONTINUE.'

```

```

ELSE
IN_PD1 = CTOD('01/01/01')
IN_PD2 = CTOD('01/01/01')
IN_TUITION = 0.00
IN_RESTAT = ' '
IN_BOOKFEE = 0.00
IN_FTDAY = 0
IN_ATPDAY = 0
IN_FSPDAY = 0

```

```

* Allow additional Pay records to be added by highlighting the next *
* available line and accepting user inputs for that record. Continue *
* the loop until user enters an <N> in the ADD field. *

```

```

DO WHILE ((ADD_MORE) .AND. (LINE_NUM <= 22))
@ 23, 0
@ 23, 0 SAY "ENTER 'Y' IN ADD FIELD TO ADD PAY RECORD. ENTER 'N'";
+ " IN ADD FIELD TO CANCEL ADD."
@ LINE_NUM, 1 GET ADD_MORE PICTURE 'Y'
@ LINE_NUM, 5 SAY LTRIM(STR(DISP_LINE))
@ LINE_NUM,10 GET IN_PD1
@ LINE_NUM,22 GET IN_PD2
@ LINE_NUM,33 GET IN_TUITION PICTURE '9999.99'
@ LINE_NUM,45 GET IN_RESTAT PICTURE '!'
@ LINE_NUM,52 GET IN_BOOKFEE PICTURE '999.99'
@ LINE_NUM,62 GET IN_FTDAY PICTURE '99'
@ LINE_NUM,69 GET IN_ATPDAY PICTURE '99'
@ LINE_NUM,76 GET IN_FSPDAY PICTURE '99'
CLEAR TYPEAHEAD

```

```

* Accept user inputs for the new Pay record. *

```

```

READ

```

```

*
```

```

IF (ADD_MORE)
  IF (IN_PD2 >= IN_PD1)
    IF (IN_PD1 > END_DATE)

```

```

* Add a new record to the file and fill it with the *
* validated input. *

```

```

APPEND BLANK
REPLACE SSAN WITH IN_SSAN
REPLACE PAY_DATE1 WITH IN_PD1
REPLACE PAY_DATE2 WITH IN_PD2
REPLACE TUITION WITH IN_TUITION
REPLACE RES_STATUS WITH IN_RESTAT
REPLACE BOOK_FEES WITH IN_BOOKFEE
REPLACE FT_DAYS WITH IN_FTDAY
REPLACE ATP_DAYS WITH IN_ATPDAY
REPLACE FSP_DAYS WITH IN_FSPDAY
@ LINE_NUM, 1 SAY ' '
@ LINE_NUM, 5 SAY LTRIM(STR(DISP_LINE))
@ LINE_NUM,10 SAY PAY_DATE1
@ LINE_NUM,22 SAY PAY_DATE2
@ LINE_NUM,33 SAY TUITION
@ LINE_NUM,45 SAY RES_STATUS
@ LINE_NUM,52 SAY BOOK_FEES
@ LINE_NUM,62 SAY FT_DAYS
@ LINE_NUM,69 SAY ATP_DAYS
@ LINE_NUM,76 SAY FSP_DAYS
END_DATE = PAY_DATE2
LINE_NUM = LINE_NUM + 1
DISP_LINE = DISP_LINE + 1
IN_PD1 = CTOD('01/01/01')

```

```

        IN_PD2      = CTOD('01/01/01')
        IN_TUITION  = 0.00
        IN_RESTAT   = ' '
        IN_BOOKFEE  = 0.00
        IN_FTDAY    = 0
        IN_ATPDAY   = 0
        IN_FSPDAY   = 0
        @ 23, 0
    ELSE
        @ 23, 0
        ? CHR(7)
        @ 23, 0 SAY 'BEGINNING PAY DATE < OR = LAST ENDING PAY' ;
                + ' DATE.  PRESS ANY KEY & TRY AGAIN.'
        WAIT ''
    ENDIF
ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 0 SAY 'ENDING PAY DATE < BEGINNING PAY DATE.' ;
            + ' PRESS ANY KEY & TRY AGAIN.'
    WAIT ''
ENDIF
ENDIF
ENDDO
ENDIF
DO RCIS_HDR
*
RETURN

```



```

*-----*
*                               EDIT_REC                               *
*-----*
*
* SUMMARY:
*
* The EDIT_REC procedure is used to update system records. The edit*
* form screens let the user type over previous entries. During edit*
* ing, the user can abort any changes and restore the record to its*
* initial state by pressing the <Esc> key. The system prevents in-*
* advertant deletion of records by "recalling" all records marked *
* for deletion. If a non-unique access key (common Last Name) has *
* been entered, the system will advise you to reenter a unique key *
* for the desired record.
*
*
* CALLED PROCEDURES:
*
* Procedure Name      Location
*-----
* DB3_ERR             RCIS_P2.PRG
* SET_UP              RCIS_P2.PRG
* INIT_DB             RCIS_P2.PRG
* EDIT_SSAN           RCIS_P2.PRG
* HGHT_CHK            RCIS_P2.PRG
* RCIS_HDR            RCIS_P2.PRG
* EDIT_PAY            RCIS_P2.PRG
* ERR_NF              RCIS_P2.PRG
* M_PROMPT            RCIS_P2.PRG
*
*-----*

```

# ``` PROCEDURE EDIT_REC ```

```

*
ON ERROR DO DB3_ERR WITH ERROR(), MESSAGE()
M_CHOICE = .T.
FIRST_TIME = .T.

* vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvv *
* Loop until user chooses to terminate this edit function mode.      *

DO WHILE (M_CHOICE)
DO SET_UP

* If the user has pressed the <Esc> key, exit this function and *
* return to the select function menu.                             *

IF (QUIT_KEY)
EXIT
ENDIF
@ 22, 0
@ 23, 0
@ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
DO INIT_DB

* If the Master file is empty, automatically exit this function and *
* return to the select function menu.                             *

```

```

IF (EMPTY_M)
  @ 22, 0
  ? CHR(7)
  @ 23, 4 SAY 'MASTER FILE IS EMPTY.  PRESS ANY KEY TO CONTINUE.'
  WAIT ''
  EXIT
ENDIF

* If the Pay file is empty and a Pay record has been selected for *
* processing, automatically exit this function and return to the *
* select function menu. *

IF (R_SELECT = 'I' .AND. EMPTY_P)
  @ 22, 0
  ? CHR(7)
  @ 23, 7 SAY 'PAY FILE IS EMPTY.  PRESS ANY KEY TO CONTINUE.'
  WAIT ''
  EXIT
ENDIF
SELECT 1

* If the user doesn't enter the primary key (IN_SSAN), use *
* the secondary key value (T_FOR_STR) which is composed of *
* the cadet's first and/or middle and/or last name. *

IF (LEN(LTRIM(IN_SSAN)) = 0)
  SET FILTER TO &T_FOR_STR

  * Issue dBASE III PLUS command to go to the first record in the *
  * file which matches the secondary key value. *

  GOTO TOP
ELSE
  SET FILTER TO &FILT_STR

  * Issue dBASE III PLUS command to go to the record which matches *
  * the primary key value. *

  SEEK IN_SSAN
ENDIF
DO CASE

  * If a matching Master record is found, set up the screen format *
  * and prepare all files required to process the record display. *

  CASE .NOT. EOF()
    IN_SSAN = SSAN
    IN_FNAM = F_NAME
    IN_MNAM = M_NAME
    IN_LNAM = L_NAME
    DO CASE

      * If the user has selected to process a Master record, *
      * issue the dBASE III PLUS commands that coordinate the *

```

\* interaction between the supporting files and the main \*  
\* file. \*

CASE R\_SELECT = 'H'  
REC\_NUM = RECNO()  
DO EDIT\_SSAN  
ASCL\_B4 = AS\_CLASS

\*

SELECT 3  
SEEK ASCL\_B4  
IF (.NOT. EOF())  
CLAS\_NUM = STR(AS\_CL\_TOT,3)  
ELSE  
CLAS\_NUM = ' ? '  
ENDIF

\*

SELECT 1  
GOTO REC\_NUM  
SET FORMAT TO &M\_FORM\_STR  
SET SCOREBOARD ON  
SET ESCAPE ON  
CLEAR TYPEAHEAD  
SET CONFIRM OFF

\* Issue 'CHANGE' command to display the record data. \*

CHANGE  
SET CONFIRM ON  
GOTO REC\_NUM  
IF PERM\_STRT = 'SAME'  
REPLACE PERM\_STRT WITH LOCAL\_STRT  
REPLACE PERM\_CITY WITH LOCAL\_CITY  
REPLACE PERM\_STAT WITH 'AZ'  
REPLACE PERM\_ZIP WITH LOCAL\_ZIP  
T\_PHON = '602' + LOCAL\_PHON  
REPLACE PERM\_PHON WITH T\_PHON

ENDIF

REPLACE WPSS WITH ((DC\_RTNG\*DCR\_VAL)+(CUM\_GPA\*100.00\*GPA\_VAL);  
+ (SAT\_CUM\*SAT\_VAL)+(AFOQT\_AA\*AA\_VAL)+(AFOQT\_QUAN\*QUAN\_VAL);  
+ (AFOQT\_VERB\*VERB\_VAL))

DO HGHT\_CHK  
IF (AS\_CLASS <> ASCL\_B4)  
CLAS\_VAL = AS\_CLASS  
SET FILTER TO  
COUNT FOR AS\_CLASS = CLAS\_VAL TO CLAS\_TOT

\*

SELECT 3  
SEEK CLAS\_VAL  
IF (.NOT. EOF())  
REPLACE AS\_CL\_TOT WITH CLAS\_TOT  
ENDIF

\*

SELECT 1  
ENDIF  
GOTO REC\_NUM

```

* If Master record was inadvertantly deleted, recall *
* it back to a current status. *

IF DELETED()
  RECALL RECORD REC_NUM
ENDIF
DO RCIS_HDR

* If the Pay file is not empty, invoke the proce- *
* dures which will give the user the opportunity *
* to view any Pay records associated with the se- *
* lected Master record. *

IF (.NOT. EMPTY_P)
  DO EDIT_PAY
  IF (VP_CHOICE)
    IF (M_CHOICE)
      LOOP
    ELSE
      EXIT
    ENDIF
  ENDIF
ENDIF

* If the user has selected to process a Pay record, *
* invoke the EDIT_PAY procedure and process its *
* return response. *

CASE R_SELECT = 'I'
  DO EDIT_PAY
  IF (M_CHOICE)
    LOOP
  ELSE
    EXIT
  ENDIF
ENDCASE

* If no matching Master record is found, give the user the option *
* to try again or to terminate this function. *

CASE EOF()
  @ 22, 0
  @ 23, 4 SAY 'MASTER '
  DO ERR_NF
  IF (M_CHOICE)
    LOOP
  ELSE
    EXIT
  ENDIF
ENDCASE

* Give the user the opportunity to execute this function again. *

DO M_PROMPT

```

ENDDO

\* Close the database files used in this function. \*

SELECT 3

USE

SELECT 2

USE

SELECT 1

USE

CLOSE FORMAT

\*

F\_PARA = STUFF(F\_PARA,1,1,'C')

@ 21, 0

ON ERROR

\*

RETURN

```

*-----*
*                                EDIT_SSAN                                *
*-----*
*
* SUMMARY:
*
*   The EDIT_SSAN procedure allows the user to change the primary key*
*   (SSAN). This procedure is controlled by the EDIT_REC procedure *
*   and is only invoked after the controlling procedure has located *
*   the Master record. The primary(SSAN) and secondary (F_NAME, *
*   M_NAME & L_NAME) keys for the current record will be displayed on*
*   the screen and the system will allow the user to change the pri- *
*   mary key if desired. This procedure is only invoked when the user*
*   has selected a Master record to edit. If the primary key is *
*   changed, this procedure will also check the Pay record file for *
*   any corresponding Pay records and change them to match the new *
*   key. The system checks to see if the new key already exists *
*   before it makes any changes.
*
*
* VARIABLE DECLARATIONS:
*
*
*   Variable Name      Status      Purpose
*   -----
*   NEW_SSAN           LOCAL        Used to store the new value for the
*                                   primary key.
*
*   ES_CHOICE          LOCAL        Boolean flag which indicates whether the
*                                   user wants to change the primary key
*                                   (SSAN).
*
*   DONE              LOCAL        Boolean flag which indicates whether the
*                                   procedure has completed processing the
*                                   changes or has encountered an error.
*
*-----*

```

# PROCEDURE EDIT\_SSAN

```

*
ES_CHOICE = .F.
@ 22, 0
@ 23, 0
? CHR(7)
@ 23, 4 SAY "MASTER RECORD FOUND. DO YOU WANT TO CHANGE THIS CADET'S SSAN" ;
        + " [Y/N]? " GET ES_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
IF (ES_CHOICE)
    DONE = .F.
    NEW_SSAN = '

* Continue loop until user enters a valid primary key change or a *
* valid exit sequence.
*

DO WHILE (.NOT. DONE)

```

```

@ 18, 0 CLEAR TO 24,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'
@ 19, 5 SAY 'First Name'
@ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
@ 20, 4 SAY 'Middle Name'
@ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
@ 21, 6 SAY 'Last Name'
@ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
@ 18,35 SAY 'New SSAN' GET NEW_SSAN PICTURE '@R 999-99-9999'
@ 20,35 SAY 'Enter New SSAN or '
@ 21,35 SAY 'Press ESC to Continue.'
CLEAR TYPEAHEAD

```

```

* Accept user's input for primary key change. *

```

```

READ
DONE = .T.

```

```

* If the input wasn't null and it wasn't equal to the existing *
* one, then continue with the following: *

```

```

IF (LEN(LTRIM(NEW_SSAN)) <> 0) .AND. (SSAN <> NEW_SSAN)
DO SSAN_CHK WITH NEW_SSAN

```

```

* Continue if input syntax is correct. *

```

```

IF (.NOT. BAD_SSAN)
SET FILTER TO
SEEK NEW_SSAN

```

```

* Continue if new input key doesn't already exist. *

```

```

IF (EOF())
IF (.NOT. (EMPTY_P))
SELECT 2
SET FILTER TO &FILT_STR
SEEK IN_SSAN

```

```

* Continue loop until all associated Pay records have *
* been reassigned the new key value. *

```

```

DO WHILE (.NOT. EOF())
REPLACE SSAN WITH NEW_SSAN
SEEK IN_SSAN
ENDDO

```

```

ENDIF
SELECT 1
GOTO REC_NUM

```

```

* Reassign selected Master record with new key value. *

```

```

REPLACE SSAN WITH NEW_SSAN
IN_SSAN = SSAN
IN_FNAM = F_NAME

```

```

        IN_MNAM = M_NAME
        IN_LNAM = L_NAME
    ELSE
        @ 23, 0
        ? CHR(7)
        @ 23, 0 SAY 'SSAN ALREADY ASSIGNED TO ANOTHER RECORD.  PRESS';
                + ' ANY KEY AND TRY AGAIN.'
        WAIT ''
        DONE = .F.
        LOOP
    ENDIF
ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 0 SAY 'SSAN FIELD MUST HAVE NINE (9) DIGITS.  PRESS ANY';
            + ' KEY AND TRY AGAIN.'
    WAIT ''
    @ 23, 0
    DONE = .F.
    LOOP
ENDIF
ENDDO
ENDIF
SET FILTER TO &FILT_STR
*
RETURN

```



```

*-----*
*                               EDIT_PAY                               *
*-----*
*
* SUMMARY:
*   The EDIT_PAY procedure allows the user to update Pay records al- *
*   ready on file. This procedure is controlled by the EDIT_REC pro- *
*   cedure and is only invoked after the controlling procedure has lo- *
*   cated a corresponding Master record. This procedure edit checks *
*   the newly entered pay date periods to ensure they don't overlap *
*   and it allows the user to view all Pay records on the same screen *
*   (16 maximum). The user is asked to enter the number which corres- *
*   ponds to the record they want to update and the system highlights *
*   the selected record. This procedure is terminated when the user *
*   enters a <0> in the prompt field.
*
* CALLED PROCEDURES:
*
*   Procedure Name      Location
*   -----
*   ERR_NF              RCIS_P2.PRG
*   VP_PROMPT           RCIS_P2.PRG
*   INIT_SAV            RCIS_P2.PRG
*   SAV_RECS            RCIS_P2.PRG
*   EDT_LINE            RCIS_P2.PRG
*   RCIS_HDR            RCIS_P2.PRG
*   M_PROMPT            RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   ED_REC_CHR         LOCAL      Used to store the user's input record
*                                number.
*
*   ED_REC_NUM         LOCAL      Used to store the numeric equivalent of
*                                ED_REC_CHR.
*
*-----*

```

# PROCEDURE EDIT\_PAY

```

*
SELECT 2
SET FILTER TO &FILT_STR

* Issue dBASE III PLUS command to go to the record which matches *
* the primary key value. *

SEEK IN_SSAN
DO CASE

* If no matching Pay records are found, give the user the option *
* to try again or to terminate this function. *

CASE EOF()

```

```

IF (R_SELECT = 'I')
  @ 22, 0
  @ 23, 7 SAY 'PAY '
  DO ERR_NF
ENDIF

```

```

* If matching Pay records are found, build Edit Pay records screen *
* and display all the current associated Pay records. *

```

```

CASE .NOT. EOF()
  VP_CHOICE = .F.
  IF (R_SELECT = 'H')
    DO VP_PRMP
  ENDIF
  IF (VP_CHOICE) .OR. (R_SELECT = 'I')
    SET SCOREBOARD ON
    SET ESCAPE ON
    CLEAR TYPEAHEAD
    @ 1, 0 TO 3,79 DOUBLE
    @ 2,25 SAY 'INDIVIDUAL CADET DATA - PAY INFORMATION'
    INITIALS = LEFT(IN_FNAM,1)+' '+LEFT(IN_MNAM,1)
    @ 2, 2 SAY TRIM(LEFT(IN_LNAM,10))+' '+INITIALS
    @ 4, 0 SAY '      REC      BEGINNING      ENDING      RESID      ';
    + 'BOOK      FT      ATP      FSP '
    @ 5, 0 SAY '      #      PAY DATE      PAY DATE      TUITION (I OR O) ';
    + 'FEES      DAYS      DAYS      DAYS'
    DO INIT_SAV
    DISP_LINE = 1
    LINE_NUM = 6

```

```

* Continue loop until all associated Pay records have been *
* displayed. *

```

```

DO WHILE ((.NOT. EOF()) .AND. (LINE_NUM <= 22))
  @ LINE_NUM, 5 SAY LTRIM(STR(DISP_LINE))
  @ LINE_NUM,10 SAY PAY_DATE1
  @ LINE_NUM,22 SAY PAY_DATE2
  @ LINE_NUM,33 SAY TUITION
  @ LINE_NUM,45 SAY RES_STATUS
  @ LINE_NUM,52 SAY BOOK_FEES
  @ LINE_NUM,62 SAY FT_DAYS
  @ LINE_NUM,69 SAY ATP_DAYS
  @ LINE_NUM,76 SAY FSP_DAYS
  DO SAV_RECS
  DISP_LINE = DISP_LINE + 1
  LINE_NUM = LINE_NUM + 1
  SKIP

```

```

ENDDO
ED_REC_NUM = 1

```

```

* Continue loop until user enters the termination value of *
* <0> in the response field. *

```

```

DO WHILE (ED_REC_NUM <> 0)
  ED_REC_CHR = '0 '

```

```

@ 23, 0
@ 23, 0 SAY 'ENTER THE REC# OF PAY RECORD TO BE EDITED (OR 0';
      + ' TO EXIT) -> ' GET ED_REC_CHR PICTURE '99'
CLEAR TYPEAHEAD
READ
ED_REC_CHR = LTRIM(RTRIM(ED_REC_CHR))
ED_REC_NUM = INT(VAL(ED_REC_CHR))
IF (ED_REC_NUM <> 0)
  DO EDT_LINE
  IF LINE_NUM > 0
    @ LINE_NUM, 5 SAY LTRIM(STR(ED_REC_NUM))
    @ LINE_NUM, 10 SAY PAY_DATE1
    @ LINE_NUM, 22 SAY PAY_DATE2
    @ LINE_NUM, 33 SAY TUITION
    @ LINE_NUM, 45 SAY RES_STATUS
    @ LINE_NUM, 52 SAY BOOK_FEES
    @ LINE_NUM, 62 SAY FT_DAYS
    @ LINE_NUM, 69 SAY ATP_DAYS
    @ LINE_NUM, 76 SAY FSP_DAYS
  ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 0 SAY 'ENTERED AN INVALID REC#. PRESS ANY KEY &' ;
      + ' TRY AGAIN.'
    WAIT ''
  ENDIF
ENDIF
ENDDO
DO RCIS_HDR

* Give the user the opportunity to execute this function again.*

DO M_PROMPT
ENDIF
ENDCASE
*
RETURN

```

```

*-----*
*                               EDT_LINE                               *
*-----*
*
* SUMMARY:
*
*   The EDT_LINE procedure searches through previously saved record
*   number values and locates the Pay records which are before and
*   after current record. It saves the date boundaries from those
*   records so the system can ensure that the updates do not cause
*   any of the pay periods to overlap. This procedure is also
*   controlled by the EDIT_REC procedure.
*
* CALLED PROCEDURES:
*
*                               Procedure Name           Location
*                               -----
*                               ED_GETS                  RCIS_P2.PRG
*
*-----*

```

# ``` PROCEDURE EDT_LINE ```

```

*
LINE_NUM = 0
LOW_DATE = CTOD ('01/01/01')
HIGH_DATE = CTOD ('12/31/99')
DO CASE
  CASE ED_REC_NUM = 1
    IF SAV_REC1 > 0
      IF SAV_REC2 > 0
        GOTO SAV_REC2
        HIGH_DATE = PAY_DATE1
      ENDIF
      LINE_NUM = ED_REC_NUM + 5
      GOTO SAV_REC1
      DO ED_GETS
    ENDIF
  CASE ED_REC_NUM = 2
    IF SAV_REC2 > 0
      IF SAV_REC1 > 0
        GOTO SAV_REC1
        LOW_DATE = PAY_DATE2
      ENDIF
      IF SAV_REC3 > 0
        GOTO SAV_REC3
        HIGH_DATE = PAY_DATE1
      ENDIF
      LINE_NUM = ED_REC_NUM + 5
      GOTO SAV_REC2
      DO ED_GETS
    ENDIF
  CASE ED_REC_NUM = 3
    IF SAV_REC3 > 0
      IF SAV_REC2 > 0
        GOTO SAV_REC2
        LOW_DATE = PAY_DATE2

```

```

ENDIF
IF SAV_REC4 > 0
    GOTO SAV_REC4
    HIGH_DATE = PAY_DATE1
ENDIF
LINE_NUM = ED_REC_NUM + 5
GOTO SAV_REC3
DO ED_GETS
ENDIF
CASE ED_REC_NUM = 4
IF SAV_REC4 > 0
    IF SAV_REC3 > 0
        GOTO SAV_REC3
        LOW_DATE = PAY_DATE2
    ENDIF
    IF SAV_REC5 > 0
        GOTO SAV_REC5
        HIGH_DATE = PAY_DATE1
    ENDIF
    LINE_NUM = ED_REC_NUM + 5
    GOTO SAV_REC4
    DO ED_GETS
ENDIF
CASE ED_REC_NUM = 5
IF SAV_REC5 > 0
    IF SAV_REC4 > 0
        GOTO SAV_REC4
        LOW_DATE = PAY_DATE2
    ENDIF
    IF SAV_REC6 > 0
        GOTO SAV_REC6
        HIGH_DATE = PAY_DATE1
    ENDIF
    LINE_NUM = ED_REC_NUM + 5
    GOTO SAV_REC5
    DO ED_GETS
ENDIF
CASE ED_REC_NUM = 6
IF SAV_REC6 > 0
    IF SAV_REC5 > 0
        GOTO SAV_REC5
        LOW_DATE = PAY_DATE2
    ENDIF
    IF SAV_REC7 > 0
        GOTO SAV_REC7
        HIGH_DATE = PAY_DATE1
    ENDIF
    LINE_NUM = ED_REC_NUM + 5
    GOTO SAV_REC6
    DO ED_GETS
ENDIF
CASE ED_REC_NUM = 7
IF SAV_REC7 > 0
    IF SAV_REC6 > 0
        GOTO SAV_REC6

```

```

        LOW_DATE = PAY_DATE2
    ENDIF
    IF SAV_REC8 > 0
        GOTO SAV_REC8
        HIGH_DATE = PAY_DATE1
    ENDIF
    LINE_NUM = ED_REC_NUM + 5
    GOTO SAV_REC7
    DO ED_GETS
ENDIF
CASE ED_REC_NUM = 8
    IF SAV_REC8 > 0
        IF SAV_REC7 > 0
            GOTO SAV_REC7
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC9 > 0
            GOTO SAV_REC9
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC8
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 9
    IF SAV_REC9 > 0
        IF SAV_REC8 > 0
            GOTO SAV_REC8
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC10 > 0
            GOTO SAV_REC10
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC9
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 10
    IF SAV_REC10 > 0
        IF SAV_REC9 > 0
            GOTO SAV_REC9
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC11 > 0
            GOTO SAV_REC11
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC10
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 11
    IF SAV_REC11 > 0
        IF SAV_REC10 > 0

```

```

        GOTO SAV_REC10
        LOW_DATE = PAY_DATE2
    ENDIF
    IF SAV_REC12 > 0
        GOTO SAV_REC12
        HIGH_DATE = PAY_DATE1
    ENDIF
    LINE_NUM = ED_REC_NUM + 5
    GOTO SAV_REC11
    DO ED_GETS
ENDIF
CASE ED_REC_NUM = 12
    IF SAV_REC12 > 0
        IF SAV_REC11 > 0
            GOTO SAV_REC11
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC13 > 0
            GOTO SAV_REC13
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC12
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 13
    IF SAV_REC13 > 0
        IF SAV_REC12 > 0
            GOTO SAV_REC12
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC14 > 0
            GOTO SAV_REC14
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC13
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 14
    IF SAV_REC14 > 0
        IF SAV_REC13 > 0
            GOTO SAV_REC13
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC15 > 0
            GOTO SAV_REC15
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC14
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 15
    IF SAV_REC15 > 0

```

```

        IF SAV_REC14 > 0
            GOTO SAV_REC14
            LOW_DATE = PAY_DATE2
        ENDIF
        IF SAV_REC16 > 0
            GOTO SAV_REC16
            HIGH_DATE = PAY_DATE1
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC15
        DO ED_GETS
    ENDIF
CASE ED_REC_NUM = 16
    IF SAV_REC16 > 0
        IF SAV_REC15 > 0
            GOTO SAV_REC15
            LOW_DATE = PAY_DATE2
        ENDIF
        LINE_NUM = ED_REC_NUM + 5
        GOTO SAV_REC16
        DO ED_GETS
    ENDIF
ENDCASE
*
RETURN

```



```

*-----*
*                                ED_GETS                                *
*-----*
*
* SUMMARY:
*      The ED_GETS procedure highlights the record selected for the
*      update, accepts the user's changes and executes the commands
*      which actually change the database files.
*
* VARIABLE DECLARATIONS:
*
*      Variable Name      Status      Purpose
*      -----
*      IN_SSAN            LOCAL      Used to store user update inputs.
*      IN_PD1              "         "
*      IN_PD2              "         "
*      IN_TUITION          "         "
*      IN_RESTAT           "         "
*      IN_BOOKFEE          "         "
*      IN_FTDAY            "         "
*      IN_ATPDAY           "         "
*      IN_FSPDAY           "         "
*-----*

```

PROCEDURE ED\_GETS

```

*
* PRIVATE BAD_ENTRY
*
BAD_ENTRY = .T.
IN_PD1     = PAY_DATE1
IN_PD2     = PAY_DATE2
IN_TUITION = TUITION
IN_RESTAT  = RES_STATUS
IN_BOOKFEE = BOOK_FEES
IN_FTDAY   = FT_DAYS
IN_ATPDAY  = ATP_DAYS
IN_FSPDAY  = FSP_DAYS

* Continue loop until all changes for the selected Pay record have been *
* validated and the entire entry is determined to be a "Good Entry".  *

DO WHILE (BAD_ENTRY)
  @ LINE_NUM, 5 SAY LTRIM(STR(ED_REC_NUM))
  @ LINE_NUM, 10 GET IN_PD1
  @ LINE_NUM, 22 GET IN_PD2
  @ LINE_NUM, 33 GET IN_TUITION PICTURE '9999.99'
  @ LINE_NUM, 45 GET IN_RESTAT  PICTURE '!'
  @ LINE_NUM, 52 GET IN_BOOKFEE PICTURE '999.99'
  @ LINE_NUM, 62 GET IN_FTDAY   PICTURE '99'
  @ LINE_NUM, 69 GET IN_ATPDAY  PICTURE '99'
  @ LINE_NUM, 76 GET IN_FSPDAY  PICTURE '99'
*
  CLEAR TYPEAHEAD

```

```

* Accept the user's input for this Pay record change.  *

READ
*
IF IN_PD1 <= IN_PD2
  IF IN_PD1 > LOW_DATE
    IF IN_PD2 < HIGH_DATE

      * Update the Pay record with the validated information.  *

      REPLACE SSAN      WITH IN_SSAN
      REPLACE PAY_DATE1 WITH IN_PD1
      REPLACE PAY_DATE2 WITH IN_PD2
      REPLACE TUITION   WITH IN_TUITION
      REPLACE RES_STATUS WITH IN_RESTAT
      REPLACE BOOK_FEES WITH IN_BOOKFEE
      REPLACE FT_DAYS   WITH IN_FTDAY
      REPLACE ATP_DAYS  WITH IN_ATPDAY
      REPLACE FSP_DAYS  WITH IN_FSPDAY
      BAD_ENTRY = .F.
    ELSE
      @ 23, 0
      ? CHR(7)
      @ 23, 0 SAY 'ENDING PAY DATE > NEXT BEGINNING PAY DATE.';
              + ' PRESS ANY KEY & TRY AGAIN.'
      WAIT ''
    ENDIF
  ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 0 SAY 'BEGINNING PAY DATE < PREVIOUS ENDING PAY DATE.';
            + ' PRESS ANY KEY & TRY AGAIN.'
    WAIT ''
  ENDIF
ELSE
  @ 23, 0
  ? CHR(7)
  @ 23, 0 SAY 'BEGINNING PAY DATE > OR = ENDING PAY DATE.';
          + ' PRESS ANY KEY & TRY AGAIN.'
  WAIT ''
ENDIF
ENDDO
*
RETURN

```

```

*-----*
*                                DEL_REC                                *
*-----*
*
* SUMMARY:
*      The DEL_REC procedure allows the user to delete records from the
*      system.  The user is asked to confirm that the record selected
*      should be deleted.  For Master records, all subordinate Pay
*      records are also deleted.
*
* CALLED PROCEDURES:
*
*      Procedure Name      Location
*      -----
*      DB3_ERR             RCIS_P2.PRG
*      SET_UP              RCIS_P2.PRG
*      INIT_DB             RCIS_P2.PRG
*      RCIS_HDR            RCIS_P2.PRG
*      D_PROMPT            RCIS_P2.PRG
*      DEL_PAY             RCIS_P2.PRG
*      ERR_NF              RCIS_P2.PRG
*      M_PROMPT            RCIS_P2.PRG
*
*-----*

```

# ``` PROCEDURE DEL_REC ```

```

*
ON ERROR DO DB3_ERR WITH ERROR(), MESSAGE()
M_CHOICE = .T.
FIRST_TIME = .T.

* vvvvvvvvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvvvvvvvvvv *
* Loop until user chooses to terminate this delete function mode. *

DO WHILE (M_CHOICE)
DO SET_UP

* If the user has pressed the <Esc> key, exit this function and *
* return to the select function menu. *

IF (QUIT_KEY)
EXIT
ENDIF
@ 22, 0
@ 23, 0
@ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
DO INIT_DB

* If the Master file is empty, automatically exit this function and *
* return to the select function menu. *

IF (EMPTY_M)
@ 22, 0
? CHR(7)
@ 23, 4 SAY 'MASTER FILE IS EMPTY. PRESS ANY KEY TO CONTINUE.'

```

```

        WAIT ''
        EXIT
    ENDIF

    * If the Pay file is empty and a Pay record has been selected for *
    * processing, automatically exit this function and return to the *
    * select function menu. *

    IF (R_SELECT = 'I' .AND. EMPTY_P)
        @ 22, 0
        ? CHR(7)
        @ 23, 7 SAY 'PAY FILE IS EMPTY.  PRESS ANY KEY TO CONTINUE.'
        WAIT ''
        EXIT
    ENDIF
    SELECT 1

    * If the user doesn't enter the primary key (IN_SSAN), use *
    * the secondary key value (T_FOR_STR) which is composed of *
    * the cadet's first and/or middle and/or last name. *

    IF (LEN(LTRIM(IN_SSAN)) = 0)
        SET FILTER TO &T_FOR_STR

        * Issue dBASE III PLUS command to go to the first record in the *
        * file which matches the secondary key value. *

        GOTO TOP
    ELSE
        SET FILTER TO &FILT_STR

        * Issue dBASE III PLUS command to go to the record which matches *
        * the primary key value. *

        SEEK IN_SSAN
    ENDIF
    DO CASE

        * If a matching Master record is found, set up the screen format *
        * and prepare all files required to process the record display. *

        CASE .NOT. EOF()
            IN_SSAN = SSAN
            IN_FNAM = F_NAME
            IN_MNAM = M_NAME
            IN_LNAM = L_NAME
            DO CASE

                * If the user has selected to process a Master record, *
                * issue the dBASE III PLUS commands that coordinate the *
                * interaction between the supporting files and the main *
                * file. *

                CASE R_SELECT = 'H'
                    REC_NUM = RECNO()

```

```

@ 22, 0
@ 23, 0
? CHR(7)
@ 23, 4 SAY 'MASTER RECORD FOUND.  PRESS ANY KEY TO' ;
      + ' VIEW RECORD.'
CLEAR TYPEAHEAD
WAIT ''
CLAS_VAL = AS_CLASS

```

\*

```

SELECT 3
SEEK CLAS_VAL
IF (.NOT. EOF())
    CT_REC_NUM = RECNO()
    CLAS_NUM = STR(AS_CL_TOT,3)
ELSE
    CLAS_NUM = ' ? '
ENDIF

```

\*

```

SELECT 1
GOTO REC_NUM
SET FORMAT TO &M_FORM_STR
SET SCOREBOARD ON
SET ESCAPE ON
CLEAR TYPEAHEAD
SET CONFIRM OFF

```

\* Issue 'CHANGE' command to display the record data. \*

```

CHANGE
SET CONFIRM ON
DO RCIS_HDR
DO D_PROMPT

```

```

* If user confirms their deletion request, then *
* delete the Master record plus any associated *
* Pay records and readjust the enrollment totals *
* relation. *

```

```

IF (P_CHOICE)
@ 23, 0
@ 23,13 SAY 'DELETING MASTER RECORD AND ';
      + 'ALL ASSOCIATED PAY RECORDS.'
GOTO REC_NUM
IF (CLAS_NUM <> ' ? ')
    CLAS_VAL = AS_CLASS
    SET FILTER TO
    COUNT FOR AS_CLASS = CLAS_VAL TO CLAS_TOT

```

\*

```

SELECT 3
GOTO CT_REC_NUM
REPLACE AS_CL_TOT WITH (CLAS_TOT - 1)

```

\*

```

SELECT 1
GOTO REC_NUM
ENDIF

```

```

DELETE
PACK

* If the Pay file is not empty, proceed to delete *
* all associated Pay records. *

IF (.NOT. EMPTY_P)
  SELECT 2
  SET FILTER TO &FILT_STR
  SEEK IN_SSAN
  DO WHILE (.NOT. EOF())
    DELETE
    SKIP
  ENDDO
  PACK
ENDIF
ENDIF
DO RCIS_HDR

* If the user has selected to process a Pay record, *
* invoke the DEL_PAY procedure and process its *
* return response. *

CASE R_SELECT = 'I'
  DO DEL_PAY
  IF (M_CHOICE)
    LOOP
  ELSE
    EXIT
  ENDIF
ENDCASE

* If no matching Master record is found, give the user the option *
* to try again or to terminate this function. *

CASE EOF()
  @ 22, 0
  @ 23, 4 SAY 'MASTER '
  DO ERR_NF
  IF (M_CHOICE)
    LOOP
  ELSE
    EXIT
  ENDIF
ENDCASE

* Give the user the opportunity to execute this function again. *

DO M_PROMPT
ENDDO

* Close the database files used in this function. *

SELECT 3
USE

```

```
SELECT 2
USE
SELECT 1
USE
CLOSE FORMAT
*
F_PARA = STUFF(F_PARA,1,1,'C')
@ 21, 0
ON ERROR
*
RETURN
```

```

*-----*
*                                DEL_PAY                                *
*-----*
*
* SUMMARY:
*
*   The DEL_PAY procedure allows the user to delete Pay records from
*   the system. This procedure is controlled by the DEL_REC proce-
*   dure and is only invoked after the controlling procedure has lo-
*   cated a corresponding Master record. This procedure allows the
*   user to view all Pay records on the same screen (16 maximum). The
*   user is asked to enter a <Y> next to each record they want to
*   "mark" for deletion. When the user is finished "marking" records
*   for deletion, they must press the <Enter> key to process their
*   input. The system bell will sound and all "marked" records will
*   be deleted.
*
* CALLED PROCEDURES:
*
*   Procedure Name      Location
*   -----
*   ERR_NF              RCIS_P2.PRG
*   INIT_SAV            RCIS_P2.PRG
*   SAV_RECS            RCIS_P2.PRG
*   INIT_FLG            RCIS_P2.PRG
*   DEL_FLGS            RCIS_P2.PRG
*   RCIS_HDR            RCIS_P2.PRG
*   M_PROMPT            RCIS_P2.PRG
*-----*

```

# PROCEDURE DEL\_PAY

```

*
SELECT 2

* Issue dBASE III PLUS command to go to the record which matches *
* the primary key value. *

SET FILTER TO &FILT_STR
SEEK IN_SSN
DO CASE

* If no matching Pay records are found, give the user the option *
* to try again or to terminate this function. *

CASE EOF()
  @ 22, 0
  @ 23, 7 SAY 'PAY '
  DO ERR_NF

* If matching Pay records are found, build Edit Pay records screen *
* and display all the current associated Pay records. *

CASE .NOT. EOF()
  SET SCOREBOARD ON
  SET ESCAPE ON

```



```

CLEAR TYPEAHEAD
@ 1, 0 TO 3,79 DOUBLE
@ 2,25 SAY 'INDIVIDUAL CADET DATA - PAY INFORMATION'
INITIALS = LEFT(IN_FNAM,1)+' '+LEFT(IN_MNAM,1)
@ 2, 2 SAY TRIM(LEFT(IN_LNAM,10))+' '+INITIALS
@ 4, 0 SAY '      REC      BEGINNING      ENDING      RESID      ' ;
      + 'BOOK      FT      ATP      FSP      '
@ 5, 0 SAY 'DEL #      PAY DATE      PAY DATE      TUITION (I OR O) ' ;
      + 'FEES      DAYS      DAYS      DAYS'

DO INIT_SAV
DISP_LINE = 1
LINE_NUM = 6

* Continue loop until all associated Pay records have been *
* displayed. *

DO WHILE ((.NOT. EOF()) .AND. (LINE_NUM <= 22))
@ LINE_NUM, 5 SAY LTRIM(STR(DISP_LINE))
@ LINE_NUM,10 SAY PAY_DATE1
@ LINE_NUM,22 SAY PAY_DATE2
@ LINE_NUM,33 SAY TUITION
@ LINE_NUM,45 SAY RES_STATUS
@ LINE_NUM,52 SAY BOOK_FEES
@ LINE_NUM,62 SAY FT_DAYS
@ LINE_NUM,69 SAY ATP_DAYS
@ LINE_NUM,76 SAY FSP_DAYS
DO SAV_RECS
DISP_LINE = DISP_LINE + 1
LINE_NUM = LINE_NUM + 1

* Issue dBASE III PLUS command to go to the next record *
* that matches the primary key value *

SKIP
ENDDO
@ 23, 0
@ 23, 7 SAY "ENTER A 'Y' IN THE DEL FIELD FOR EACH PAY RECORD";
      + " YOU WANT DELETED."
DO INIT_FLG
DO DEL_FLGS
DO RCIS_HDR

* Give the user the opportunity to execute this function again.*

DO M_PROMPT

ENDCASE
*
RETURN

```

```

*-----*
*                                     DEL_FLGS                                     *
*-----*
*
* SUMMARY:
* The DEL_FLGS procedure highlights the record deletion fields,
* processes the user's deletion requests and deletes the appro-
* priate Pay records.
*
*-----*

```

# PROCEDURE DEL\_FLGS

```

*
  LINE_NUM = 6
  IF (SAV_REC1 > 0)
    @ LINE_NUM, 1 GET FLAG_REC1 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC2 > 0)
    @ LINE_NUM, 1 GET FLAG_REC2 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC3 > 0)
    @ LINE_NUM, 1 GET FLAG_REC3 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC4 > 0)
    @ LINE_NUM, 1 GET FLAG_REC4 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC5 > 0)
    @ LINE_NUM, 1 GET FLAG_REC5 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC6 > 0)
    @ LINE_NUM, 1 GET FLAG_REC6 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC7 > 0)
    @ LINE_NUM, 1 GET FLAG_REC7 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC8 > 0)
    @ LINE_NUM, 1 GET FLAG_REC8 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC9 > 0)
    @ LINE_NUM, 1 GET FLAG_REC9 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF
  IF (SAV_REC10 > 0)
    @ LINE_NUM, 1 GET FLAG_REC10 PICTURE 'Y'
    LINE_NUM = LINE_NUM + 1
  ENDIF

```

```

IF (SAV_REC11 > 0)
  @ LINE_NUM, 1 GET FLAG_REC11 PICTURE 'Y'
  LINE_NUM = LINE_NUM + 1
ENDIF
IF (SAV_REC12 > 0)
  @ LINE_NUM, 1 GET FLAG_REC12 PICTURE 'Y'
  LINE_NUM = LINE_NUM + 1
ENDIF
IF (SAV_REC13 > 0)
  @ LINE_NUM, 1 GET FLAG_REC13 PICTURE 'Y'
  LINE_NUM = LINE_NUM + 1
ENDIF
IF (SAV_REC14 > 0)
  @ LINE_NUM, 1 GET FLAG_REC14 PICTURE 'Y'
  LINE_NUM = LINE_NUM + 1
ENDIF
IF (SAV_REC15 > 0)
  @ LINE_NUM, 1 GET FLAG_REC15 PICTURE 'Y'
  LINE_NUM = LINE_NUM + 1
ENDIF
IF (SAV_REC16 > 0)
  @ LINE_NUM, 1 GET FLAG_REC16 PICTURE 'Y'
  LINE_NUM = LINE_NUM + 1
ENDIF
*
CLEAR TYPEAHEAD

* Accept the user's Pay record deletion requests. *

READ
*
@ 23, 0
? CHR(7)
@ 23, 0 SAY "ONLY DELETING MARKED ('Y') PAY RECORDS."
*
IF (FLAG_REC1)
  GOTO SAV_REC1
DELETE
ENDIF
IF (FLAG_REC2)
  GOTO SAV_REC2
DELETE
ENDIF
IF (FLAG_REC3)
  GOTO SAV_REC3
DELETE
ENDIF
IF (FLAG_REC4)
  GOTO SAV_REC4
DELETE
ENDIF
IF (FLAG_REC5)
  GOTO SAV_REC5
DELETE
ENDIF

```

```
IF (FLAG_REC6)
  GOTO SAV_REC6
DELETE
ENDIF
IF (FLAG_REC7)
  GOTO SAV_REC7
DELETE
ENDIF
IF (FLAG_REC8)
  GOTO SAV_REC8
DELETE
ENDIF
IF (FLAG_REC9)
  GOTO SAV_REC9
DELETE
ENDIF
IF (FLAG_REC10)
  GOTO SAV_REC10
DELETE
ENDIF
IF (FLAG_REC11)
  GOTO SAV_REC11
DELETE
ENDIF
IF (FLAG_REC12)
  GOTO SAV_REC12
DELETE
ENDIF
IF (FLAG_REC13)
  GOTO SAV_REC13
DELETE
ENDIF
IF (FLAG_REC14)
  GOTO SAV_REC14
DELETE
ENDIF
IF (FLAG_REC15)
  GOTO SAV_REC15
DELETE
ENDIF
IF (FLAG_REC16)
  GOTO SAV_REC16
DELETE
ENDIF
*
PACK
*
RETURN
```

```

*-----*
*                                VIEW_REC                                *
*-----*
*
* SUMMARY:
* The VIEW_REC procedure is used to view system records. This
* procedure only allows the user to view the contents of the record*
* fields, i.e. no updating is allowed. The system prevents inadver-
* tant deletion of records by "recalling" all records marked for
* deletion. If a non-unique access key (common Last Name) has been*
* entered the system will advise you to reenter a unique key for
* the desired record.
*
* CALLED PROCEDURES:
*
* Procedure Name      Location
* -----
* DB3_ERR             RCIS_P2.PRG
* SET_UP              RCIS_P2.PRG
* INIT_DB             RCIS_P2.PRG
* RCIS_HDR            RCIS_P2.PRG
* VIEW_PAY            RCIS_P2.PRG
* ERR_NF              RCIS_P2.PRG
* M_PROMPT            RCIS_P2.PRG
*-----*

```

#### PROCEDURE VIEW\_REC

```

*
ON ERROR DO DB3_ERR WITH ERROR(), MESSAGE()
M_CHOICE = .T.
FIRST_TIME = .T.

* vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvv *
* Loop until user chooses to terminate this view function mode.    *

DO WHILE (M_CHOICE)
  DO SET_UP

  * If the user has pressed the <Esc> key, exit this function and *
  * return to the select function menu.                            *

  IF (QUIT_KEY)
    EXIT
  ENDIF
  @ 22, 0
  @ 23, 0
  @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
  DO INIT_DB

  * If the Master file is empty, automatically exit this function and *
  * return to the select function menu.                                *

  IF (EMPTY_M)
    @ 22, 0

```

```

        ? CHR(7)
        @ 23, 4 SAY 'MASTER FILE IS EMPTY.  PRESS ANY KEY TO CONTINUE.'
        WAIT ''
        EXIT
    ENDIF

    * If the Pay file is empty and a Pay record has been selected for
    * processing, automatically exit this function and return to the
    * select function menu.
    *

    IF (R_SELECT = 'I' .AND. EMPTY_P)
        @ 22, 0
        ? CHR(7)
        @ 23, 7 SAY 'PAY FILE IS EMPTY.  PRESS ANY KEY TO CONTINUE.'
        WAIT ''
        EXIT
    ENDIF
    SELECT 1

    * If the user doesn't enter the primary key (IN_SSAN), use
    * the secondary key value (T_FOR_STR) which is composed of
    * the cadet's first and/or middle and/or last name.
    *

    IF (LEN(LTRIM(IN_SSAN)) = 0)
        SET FILTER TO &T_FOR_STR

        * Issue dBASE III PLUS command to go to the first record in the
        * file which matches the secondary key value.
        *

        GOTO TOP
    ELSE
        SET FILTER TO &FILT_STR

        * Issue dBASE III PLUS command to go to the record which matches
        * the primary key value.
        *

        SEEK IN_SSAN
    ENDIF
    DO CASE

        * If a matching Master record is found, set up the screen format
        * and prepare all files required to process the record display.
        *

        CASE .NOT. EOF()
            IN_SSAN = SSAN
            IN_FNAM = F_NAME
            IN_MNAM = M_NAME
            IN_LNAM = L_NAME
            DO CASE

                * If the user has selected to process a Master record,
                * issue the dBASE III PLUS commands that coordinate the
                * interaction between the supporting files and the main
                * file.
                *

```

```

CASE R_SELECT = 'H'
  REC_NUM = RECNO()
  @ 22, 0
  @ 23, 0
  ? CHR(7)
  @ 23, 4 SAY 'MASTER RECORD FOUND.  PRESS ANY KEY TO' ;
          + ' VIEW RECORD.'
  CLEAR TYPEAHEAD
  WAIT ''
  CLAS_VAL = AS_CLASS

*

  SELECT 3
  SEEK CLAS_VAL
  IF (.NOT. EOF())
    CLAS_NUM = STR(AS_CL_TOT,3)
  ELSE
    CLAS_NUM = ' ? '
  ENDIF

*

  SELECT 1
  GOTO REC_NUM
  SET FORMAT TO &M_FORM_STR
  SET SCOREBOARD ON
  SET ESCAPE ON
  CLEAR TYPEAHEAD
  SET CONFIRM OFF

* Issue 'CHANGE' command to display the record data. *

  CHANGE
  SET CONFIRM ON
  GOTO REC_NUM

* If Master record was inadvertantly deleted, recall *
* it back to a current status. *

  IF DELETED()
    RECALL RECORD REC_NUM
  ENDIF
  DO RCIS_HDR

* If the Pay file is not empty, invoke the proce- *
* dures which will give the user the opportunity *
* to view any Pay records associated with the se- *
* lected Master record. *

  IF (.NOT. EMPTY_P)
    DO VIEW_PAY
    IF (VP_CHOICE)
      IF (M_CHOICE)
        LOOP
      ELSE
        EXIT
      ENDIF
    ENDIF
  ENDIF

```

```

                                ENDIF

                                *   If the user has selected to process a Pay record,   *
                                *   invoke the VIEW_PAY procedure and process its       *
                                *   return response.                                   *

                                CASE  R_SELECT = 'I'
                                    DO VIEW_PAY
                                    IF (M_CHOICE)
                                        LOOP
                                    ELSE
                                        EXIT
                                    ENDIF
                                ENDCASE

                                *   If no matching Master record is found, give the user the option   *
                                *   to try again or to terminate this function.             *

                                CASE  EOF()
                                    @ 22, 0
                                    @ 23, 4 SAY 'MASTER '
                                    DO ERR_NF
                                    IF (M_CHOICE)
                                        LOOP
                                    ELSE
                                        EXIT
                                    ENDIF
                                ENDCASE

                                *   Give the user the opportunity to execute this function again.   *

                                DO M_PROMPT
                                ENDDO

                                *   Close the database files used in this function.   *

                                SELECT 3
                                USE
                                SELECT 2
                                USE
                                SELECT 1
                                USE
                                CLOSE FORMAT
                                *
                                F_PARA = STUFF(F_PARA,1,1,'C')
                                @ 21, 0
                                ON ERROR
                                *
                                RETURN

```



```

*-----*
*                                TRANS_REC                                *
*-----*
*
* SUMMARY:
*   The TRANS_REC procedure is used to transfer system records between*
*   the active and the inactive relation files. The system checks the*
*   destination file to ensure that the input primary key doesn't  *
*   already exist before the transfer is allowed to proceed. Master *
*   records and all associated subordinate Pay records will be trans- *
*   ferred at the same time. The Master record is automatically dis- *
*   played to the user and the user is given the option of viewing the*
*   associated Pay records. Transfer confirmation is required before *
*   the record is copied. The system also checks to prevent inadver- *
*   tant deletion of a record.
*
* CALLED PROCEDURES:
*
*   Procedure Name      Location
*   -----
*   DB3_ERR             RCIS_P2.PRG
*   SET_UP              RCIS_P2.PRG
*   TRANS_CHK           RCIS_P2.PRG
*   INIT_DB            RCIS_P2.PRG
*   RCIS_HDR           RCIS_P2.PRG
*   VIEW_PAY           RCIS_P2.PRG
*   TQ_PROMPT          RCIS_P2.PRG
*   BLD_NDX            RCIS_P2.PRG
*   ERR_NF             RCIS_P2.PRG
*   M_PROMPT           RCIS_P2.PRG
*
*-----*

```

# ``` PROCEDURE TRANS_REC ```

```

*
ON ERROR DO DB3_ERR WITH ERROR(), MESSAGE()
M_CHOICE = .T.
FIRST_TIME = .T.

* vvvvvvvvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvv *
* Loop until user chooses to terminate this transfer function mode. *

DO WHILE (M_CHOICE)
DO SET_UP

* If the user has pressed the <Esc> key, exit this function and *
* return to the select function menu. *

IF (QUIT_KEY)
EXIT
ENDIF
@ 22, 0
@ 23, 0
@ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
DO TRANS_CHK

```

DO CASE

```
* If the input search key already exists on the target file and *
* the user wants to try again, loop back to the beginning of the *
* "Do While (M_Choice)" statement. *
```

CASE T\_PATH = 2  
LOOP

```
* If the input search key already exists on the target file and *
* the user doesn't want to try again, exit from the transfer *
* function mode. *
```

CASE T\_PATH = 3  
EXIT

ENDCASE

DO INIT\_DB

```
* If the Master file is empty, automatically exit this function and *
* return to the select function menu. *
```

IF (EMPTY\_M)

@ 22, 0

? CHR(7)

@ 23, 4 SAY 'MASTER FILE IS EMPTY. PRESS ANY KEY TO CONTINUE.'

WAIT ''

EXIT

ENDIF

SELECT 1

SET FILTER TO &FILT\_STR

```
* Issue dBASE III PLUS command to go to the record which matches *
* the primary key value. *
```

SEEK IN\_SSAN

DO CASE

```
* If a matching Master record is found, set up the screen format *
* and prepare all files required to process the record display. *
```

CASE .NOT. EOF()

REC\_NUM = RECNO()

IN\_FNAM = F\_NAME

IN\_MNAM = M\_NAME

IN\_LNAM = L\_NAME

@ 22, 0

@ 23, 0

? CHR(7)

@ 23, 4 SAY 'MASTER RECORD FOUND. PRESS ANY KEY TO VIEW RECORD.'

CLEAR TYPEAHEAD

WAIT ''

SAV\_CLAS = AS\_CLASS

SELECT 3

SEEK SAV\_CLAS

```

IF (.NOT. EOF())
  CT_REC_NUM = RECNO()
  CLAS_NUM = STR(AS_CL_TOT,3)
ELSE
  CLAS_NUM = ' ? '
ENDIF

```

\*

```

SELECT 1
GOTO REC_NUM
SET FORMAT TO &M_FORM_STR
SET SCOREBOARD ON
SET ESCAPE ON
CLEAR TYPEAHEAD
SET CONFIRM OFF

```

\* Issue 'CHANGE' command to display the record data. \*

```

CHANGE
SET CONFIRM ON
DO RCIS_HDR
GOTO REC_NUM
DELETE

```

\* If the Pay file is not empty, invoke the procedures \*  
 \* which will give the user the opportunity to view any \*  
 \* Pay records associated with the selected Master record. \*

```

IF (.NOT. EMPTY_P)
  R_SELECT = ' '
  DO VIEW_PAY
ENDIF
DO TQ_PRMP

```

\* If user reconfirms transfer request, then prepare all \*  
 \* target files for the transfer process. \*

```

IF (TQ_CHOICE)
  @ 23, 0
  @ 23, 4 SAY 'TRANSFERING MASTER RECORD AND ALL ASSOCIATED';
  + ' PAY RECORDS TO ' + DEST_FILE + ' FILE'

```

\* Close all source files while transfer is being processed.\*

```

SELECT 3
USE
SELECT 2
USE
SELECT 1
USE

```

\*

```

SELECT 1
USE &T_M_FILE

```

\* Prepare main index file and build index list for target \*  
 \* files. \*

```

IF (.NOT. FILE(T_M_NDX_F))
  INDEX ON &M_NDX_STR TO &T_M_NDX
ENDIF
DO BLD_NDX WITH T_M_NDX
SET INDEX TO &NDX_LIST
SET FILTER TO &FILT_STR

```

\* Transfer Master record from source file to target file. \*

```
APPEND FROM &M_FILE FOR SSAN = IN_SSAN
```

\* Update target file support files (tables). \*

```

IF (CLAS_NUM <> ' ? ')
  CLAS_VAL = AS_CLASS
  SET FILTER TO
  COUNT FOR AS_CLASS = CLAS_VAL TO CLAS_TOT

```

\*

```

  SELECT 3
  USE &T_CT_FILE
  IF (.NOT. FILE(T_CT_NDX_F))
    INDEX ON AS_CLASS TO &T_CT_NDX
  ENDIF
  SET INDEX TO &T_CT_NDX
  SEEK CLAS_VAL
  IF (.NOT. EOF())
    REPLACE AS_CL_TOT WITH CLAS_TOT
  ENDIF
ENDIF

```

\*

```

SELECT 2
USE &T_P_FILE
IF (RECNO() = 1) .AND. EOF() .AND. FILE(T_P_NDX_F)
  ERASE &T_P_NDX_F
ENDIF
IF (.NOT. FILE(T_P_NDX_F))
  INDEX ON &P_NDX_STR TO &T_P_NDX
ENDIF
SET INDEX TO &T_P_NDX
SET FILTER TO &FILT_STR

```

\* Transfer all associated Pay records from the source \*

\* file to the target file. \*

```
APPEND FROM &P_FILE FOR SSAN = IN_SSAN
```

\* Transfer complete. Close all target files. \*

```

SELECT 3
USE
SELECT 2
USE
SELECT 1
USE

```

ENDIF

\*

SELECT 1  
USE &M\_FILE  
DO BLD\_NDX WITH M\_NDX  
SET INDEX TO &NDX\_LIST

\* If transfer was reconfirmed, remove marked Master record \*  
\* from the source file. \*

IF (TQ\_CHOICE)  
PACK  
IF (CLAS\_NUM <> ' ? ' )  
SET FILTER TO  
COUNT FOR AS\_CLASS = SAV\_CLAS TO CLAS\_TOT

\*

SELECT 3  
USE &CT\_FILE  
IF (.NOT. FILE(CT\_NDX\_F))  
INDEX ON AS\_CLASS TO &CT\_NDX  
ENDIF  
SET INDEX TO &CT\_NDX  
GOTO CT\_REC\_NUM  
REPLACE AS\_CL\_TOT WITH CLAS\_TOT

ENDIF

ELSE

GOTO REC\_NUM

\* If transfer request was not confirmed, recall the \*  
\* Master record back to current status. \*

IF DELETED()  
RECALL RECORD REC\_NUM  
ENDIF

ENDIF

IF (.NOT. EMPTY\_P)

SELECT 2  
USE &P\_FILE  
SET INDEX TO &P\_NDX

\* If the Pay file is not empty and the transfer request \*  
\* was confirmed, remove all associated Pay records from \*  
\* the source file. If the request wasn't confirmed, \*  
\* recall all marked Pay records back to current status. \*

IF (TQ\_CHOICE)  
PACK  
ELSE  
RECALL ALL  
ENDIF

ENDIF

\* Close all source files in preparation for next process. \*

SELECT 3

```

        USE
        SELECT 2
        USE
        SELECT 1
        USE

*   If no matching Master record is found, give the user the option *
*   to try again or to terminate this function.                      *

CASE   EOF()
      @ 22, 0
      @ 23, 4 SAY 'MASTER '
      DO ERR_NF
      IF (M_CHOICE)
        LOOP
      ELSE
        EXIT
      ENDIF
ENDCASE

*   Give the user the opportunity to execute this function again.  *

DO M_PROMPT
ENDDO

*   Close the database files used in this function.  *

SELECT 3
USE
SELECT 2
USE
SELECT 1
USE
CLOSE FORMAT
*
F_PARA = STUFF(F_PARA,1,1,'C')
@ 21, 0
ON ERROR
*
RETURN

```

```

*-----*
*                                TRANS_CHK                                *
*-----*
*
* SUMMARY:                                                                *
*   The TRANS_CHK is controlled by the TRANS_REC procedure. This pro-   *
*   cedure is used to access the target file and check for any exist-   *
*   ing primary keys which match the one input by the user. If an      *
*   existing key is found, the user is advised to check their input     *
*   and try again.                                                       *
*-----*

```

# PROCEDURE TRANS\_CHK

```

*
  T_PATH = 1
  SELECT 1
  USE &T_M_FILE

  * If the target Master file is empty and the index file exists, erase *
  * the index file.                                                       *

  IF (RECNO() = 1) .AND. EOF() .AND. FILE(T_M_NDX_F)
    ERASE &T_M_NDX_F
  ELSE
    IF (.NOT. FILE(T_M_NDX_F))
      INDEX ON &M_NDX_STR TO &T_M_NDX
    ENDIF
    SET INDEX TO &T_M_NDX
    SET FILTER TO &FILT_STR
    SEEK IN_SSAN

    * If the input key value already exists on the target file, prompt *
    * the user to try again.                                             *

    IF (.NOT. EOF())
      T_PATH = 2
      @ 22, 0
      @ 23, 0
      ? CHR(7)
      M_CHOICE = .T.
      @ 23,10 SAY 'RECORD ALREADY EXISTS IN THE TARGET FILE.'
      @ 23,53 SAY 'DO YOU WANT TO TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
      CLEAR TYPEAHEAD
      READ
      @ 23,0
      IF .NOT. M_CHOICE
        T_PATH = 3
        @ 21, 0
        @ 21,33 SAY 'CLOSING FILES'
      ENDIF
    ENDIF
  ENDIF

```

\* Close the target Master file. \*

SELECT 1

USE

\*

RETURN



```

*-----*
*                                     HGHT_CHK                                     *
*-----*
* SUMMARY:
*       The HGHT_CHK procedure is used to ensure that the HEIGHT field
*       data stored in the Master record matches the primary key field in
*       the height table relation. This procedure rounds the user's input
*       height to the nearest quarter of an inch because the height table
*       relation only recognizes quarter inch increments.
*
* INVOKING PROCEDURES:
*
*       Procedure Name      Location
*       -----
*       ADD_REC             RCIS_P2.PRG
*       EDIT_REC            RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*       Variable Name      Status      Purpose
*       -----
*       HT_NUM             LOCAL       Used to store the integer portion of the
*                                     HEIGHT variable.
*
*       HT_FRAC            LOCAL       Used to store the fraction portion of the
*                                     HEIGHT variable.
*
*-----*

```

# ``` PROCEDURE HGHT_CHK ```

```

*
  HT_NUM = VAL(LEFT(STR((HEIGHT*100),4),2))
  HT_FRAC = VAL(RIGHT(STR((HEIGHT*100),4),2))/100.00
  IF (HT_FRAC <> 0.00) .AND. (HT_FRAC <> 0.25) .AND. ;
    (HT_FRAC <> 0.50) .AND. (HT_FRAC <> 0.75)
*
  IF (HT_FRAC > 0.00) .AND. (HT_FRAC < 0.13)
    HT_FRAC = 0.00
  ELSE
    IF (HT_FRAC >= 0.13) .AND. (HT_FRAC < 0.38)
      HT_FRAC = 0.25
    ELSE
      IF (HT_FRAC >= 0.38) .AND. (HT_FRAC < 0.63)
        HT_FRAC = 0.50
      ELSE
        IF (HT_FRAC >= 0.63) .AND. (HT_FRAC < 0.88)
          HT_FRAC = 0.75
        ELSE
          HT_FRAC = 0.00
          HT_NUM = HT_NUM + 1.00
        ENDIF
      ENDIF
    ENDIF
  ENDIF
  ENDIF
  ENDIF

```

```

* If the input value for the cadet's height is outside the *
* allowable range, replace the height value with zeroes *
* (this will cause the cadet's record to be flagged on the *
* weight standards report and will prompt the user to enter *
* the correct value). *

IF (HT_NUM < 58.00) .OR. (HT_NUM > 83.00)
    HT_NUM = 0.00
    HT_FRAC = 0.00
ENDIF
REPLACE HEIGHT WITH (HT_NUM + HT_FRAC)
ENDIF
*
RETURN

```



```

IF ((R_SELECT = 'H') .OR. (F_SELECT = 'L'))
  DO VP_PRMP
ENDIF

* Enter this section if the user selected a Pay record for *
* processing or if their initial selection was a Master *
* record and they chose to view any associated Pay records. *

IF (VP_CHOICE) .OR. (R_SELECT = 'I')
  SET SCOREBOARD ON
  SET ESCAPE ON
  CLEAR TYPEAHEAD
  @ 1, 0 TO 3,79 DOUBLE
  @ 2,25 SAY 'INDIVIDUAL CADET DATA - PAY INFORMATION'
  INITIALS = LEFT(IN_FNAM,1)+' '+LEFT(IN_MNAM,1)
  @ 2, 2 SAY TRIM(LEFT(IN_LNAM,10))+' '+INITIALS
  @ 4, 0 SAY '      REC      BEGINNING      ENDING      RESID      ';
      + 'BOOK      FT      ATP      FSP '
  @ 5, 0 SAY '      #      PAY DATE      PAY DATE      TUITION (I OR O) ';
      + 'FEES      DAYS      DAYS      DAYS'
  DISP_LINE = 1
  LINE_NUM = 6

  * Continue loop until all associated Pay records have been *
  * displayed. *

  DO WHILE ((.NOT. EOF()) .AND. (LINE_NUM <= 22))
    REC_NUM = RECNO()
    @ LINE_NUM, 5 SAY LTRIM(STR(DISP_LINE))
    @ LINE_NUM,10 SAY PAY_DATE1
    @ LINE_NUM,22 SAY PAY_DATE2
    @ LINE_NUM,33 SAY TUITION
    @ LINE_NUM,45 SAY RES_STATUS
    @ LINE_NUM,52 SAY BOOK_FEES
    @ LINE_NUM,62 SAY FT_DAYS
    @ LINE_NUM,69 SAY ATP_DAYS
    @ LINE_NUM,76 SAY FSP_DAYS
    DISP_LINE = DISP_LINE + 1
    LINE_NUM = LINE_NUM + 1

    * If the transfer function was selected, delete all *
    * associated Pay records after their contents has *
    * been displayed. *

    IF (F_SELECT = 'L')
      GOTO REC_NUM
      DELETE
    ENDIF

    * Issue dBASE III PLUS command to go to the next Pay *
    * record which matches the input key value. *

    SKIP
  ENDDO
  @ 23, 0 SAY 'PRESS ANY KEY TO RETURN TO MAIN SELECTION SCREEN'

```

```

WAIT ''
DO RCIS_HDR
IF (F_SELECT = 'J')

    * Give the user the opportunity to execute this function *
    * again. *

    DO M_PROMPT
ENDIF
ELSE

    * If the transfer function was selected, delete all *
    * associated Pay records without displaying their *
    * contents. *

    IF (F_SELECT = 'L')
        DO WHILE (.NOT. EOF())
            DELETE
            SKIP
        ENDDO
    ENDIF
ENDIF
ENDCASE
*
RETURN

```

```

*-----*
*                               SET_UP                               *
*-----*
*
* SUMMARY:
*       The SET_UP procedure is used to set up the string variables used
*       to identify the different source and destination database files
*       (both data and index files). All procedures in this file use
*       these strings (GLOBAL) as opposed to building their own.
*
* INVOKING PROCEDURES:
*
*       Procedure Name      Location
*       -----
*       ADD_REC             RCIS_P2.PRG
*       EDIT_REC            RCIS_P2.PRG
*       VIEW_REC            RCIS_P2.PRG
*       DEL_REC             RCIS_P2.PRG
*       TRANS_REC           RCIS_P2.PRG
*
* CALLED PROCEDURES:
*
*       Procedure Name      Location
*       -----
*       INPUT_KEY           RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*       Variable Name      Status      Purpose
*       -----
*       S_PREFIX           LOCAL      Used to store a one letter identifier for
*                                     the source files.
*
*       T_PREFIX           LOCAL      Used to store a one letter identifier for
*                                     the target files.
*
*-----*

```

# ``` PROCEDURE SET_UP ```

```

*
*   PRIVATE S_PREFIX
*   PRIVATE T_PREFIX
*

```

```

*   Initialize global boolean variables used in other procedures.  *

```

```

QUIT_KEY = .F.
EMPTY_M  = .F.
EMPTY_P  = .F.
DEL_FLAG = .F.

```

```

* All these database file string variables only need to be built once *
* for each mode.                                                         *

```

```

IF (FIRST_TIME)
  M_FILE = 'X_CDT_MS'

```

```

P_FILE = 'X_CDT_PY'
CT_FILE = 'X_CDT_CT'

* Initialize source and target file designaters. *

IF (G_SELECT = 'H')
  S_PREFIX = 'A'
  T_PREFIX = 'I'
  DEST_FILE = 'INACTIVE'
ELSE
  S_PREFIX = 'I'
  T_PREFIX = 'A'
  DEST_FILE = 'ACTIVE'
ENDIF
M_FILE = STUFF(M_FILE,1,1,LTRIM(S_PREFIX))
P_FILE = STUFF(P_FILE,1,1,LTRIM(S_PREFIX))
CT_FILE = STUFF(CT_FILE,1,1,LTRIM(S_PREFIX))
M_NDX = 'X_SSAN'
P_NDX = 'X_PAYD'
CT_NDX = 'X_ASCL'
M_NDX_STR = 'SSAN'
P_NDX_STR = 'SSAN+STR(YEAR(PAY_DATE1),4)+STR(MONTH(PAY_DATE1),2)';
          + '+STR(DAY(PAY_DATE1),2)'
FILT_STR = 'SSAN = IN_SSAN'
IF (F_SELECT >= 'J')
  M_FORM_STR = 'CDT_M_VU'
ELSE
  M_FORM_STR = 'CDT_M'
ENDIF
IF (F_SELECT = 'L')
  T_M_FILE = STUFF(M_FILE,1,1,LTRIM(T_PREFIX))
  T_P_FILE = STUFF(P_FILE,1,1,LTRIM(T_PREFIX))
  T_CT_FILE = STUFF(CT_FILE,1,1,LTRIM(T_PREFIX))
  T_M_NDX = STUFF(M_NDX,1,1,LTRIM(T_PREFIX))
  T_P_NDX = STUFF(P_NDX,1,1,LTRIM(T_PREFIX))
  T_CT_NDX = STUFF(CT_NDX,1,1,LTRIM(T_PREFIX))
  T_M_NDX_F = T_M_NDX + '.NDX'
  T_P_NDX_F = T_P_NDX + '.NDX'
  T_CT_NDX_F = T_CT_NDX + '.NDX'
ENDIF
M_NDX = STUFF(M_NDX,1,1,LTRIM(S_PREFIX))
P_NDX = STUFF(P_NDX,1,1,LTRIM(S_PREFIX))
CT_NDX = STUFF(CT_NDX,1,1,LTRIM(S_PREFIX))
M_NDX_F = M_NDX + '.NDX'
P_NDX_F = P_NDX + '.NDX'
CT_NDX_F = CT_NDX + '.NDX'
ENDIF
DO INPUT_KEY
*
RETURN

```

```

*-----*
*                               INPUT_KEY                               *
*-----*
*
* SUMMARY:
*       The INPUT_KEY procedure displays the prompts required for access
*       keys and accepts the user's input.  If a null value is returned,
*       either by pressing the <Enter> key without previously entering
*       data or by pressing the <Esc> key, the QUIT_KEY flag is set to
*       TRUE. This serves as an escape mechanism if the user had inadver-
*       tantly selected an incorrect mode.
*
* CALLED PROCEDURES:
*
*       Procedure Name          Location
*       -----
*       SSAN_CHK                RCIS_P2.PRG
*
*-----*

```

# PROCEDURE INPUT\_KEY

```

*
  DONE = .F.
  IN_FNAM = '
  IN_MNAM = '
  IN_LNAM = '
  IN_SSAN = '
  @ 18, 0 CLEAR TO 24,79
  @ 18,11 SAY 'SSAN'

  * If the selected function is not Add or Transfer, display the
  * secondary key value.
  *

  IF (F_SELECT <> 'H') .AND. (F_SELECT <> 'L')
    @ 19, 5 SAY 'First Name'
    @ 20, 4 SAY 'Middle Name'
    @ 21, 6 SAY 'Last Name'
    @ 22,36 SAY ' OR Name.'
  ENDIF
  @ 22, 4 SAY "Enter Cadet's Social Security #"

  * Continue loop until the user enters a valid response or until they
  * enter an exit sequence.
  *

  DO WHILE (.NOT. DONE)
    @ 18,16 GET IN_SSAN PICTURE '@R 999-99-9999'

    * If the selected function is not Add or Transfer, allow the
    * user to specify a secondary key value for the search.
    *

    IF (F_SELECT <> 'H') .AND. (F_SELECT <> 'L')
      @ 19,16 GET IN_FNAM PICTURE '!!!!!!!!!!!!!!'
      @ 20,16 GET IN_MNAM PICTURE '!!!!!!!!!!!!!!'
      @ 21,16 GET IN_LNAM PICTURE '!!!!!!!!!!!!!!'
    ENDIF
  
```



\* Accept user's input key values. \*

READ  
CLEAR TYPEAHEAD

\*

DONE = .T.

\* If the user doesn't enter a value for the primary key, build the \*  
\* filter string variable from the secondary key value inputs. \*

```
IF (LEN(LTRIM(IN_SSAN)) = 0)
  IF (F_SELECT <> 'H') .AND. (F_SELECT <> 'L')
    T_FOR_STR = ''
    IF (LEN(LTRIM(IN_FNAM)) > 0)
      T_FOR_STR = 'F_NAME =' + ''' + IN_FNAM + '''
    ENDIF
    IF (LEN(LTRIM(IN_MNAM)) > 0)
      IF (LEN(T_FOR_STR) > 0)
        T_FOR_STR = T_FOR_STR + '.AND.M_NAME =' + ''' + IN_MNAM + '''
      ELSE
        T_FOR_STR = 'M_NAME =' + ''' + IN_MNAM + '''
      ENDIF
    ENDIF
    IF (LEN(LTRIM(IN_LNAM)) > 0)
      IF (LEN(T_FOR_STR) > 0)
        T_FOR_STR = T_FOR_STR + '.AND.L_NAME =' + ''' + IN_LNAM + '''
      ELSE
        T_FOR_STR = 'L_NAME =' + ''' + IN_LNAM + '''
      ENDIF
    ENDIF
  ENDIF
```

\* If the secondary key value is being used, check the file for \*  
\* duplicate records associated with that input value. \*

```
IF (LEN(T_FOR_STR) > 0)
  SELECT 1
  USE &M_FILE
  COUNT FOR &T_FOR_STR TO REC_CNT
  IF (REC_CNT > 1)
    @ 22, 0
    @ 23, 0
    ? CHR(7)
    @ 23, 0 SAY 'NAME ASSIGNED TO MORE THAN ONE RECORD (ENTER';
              + ' SSAN). PRESS ANY KEY & TRY AGAIN.'
    WAIT ''
    DONE = .F.
    LOOP
  ENDIF
ELSE
  QUIT_KEY = .T.
ENDIF
ELSE
  QUIT_KEY = .T.
ENDIF
```

```

ELSE
  DO SSAN_CHK WITH IN_SSAN

  * If the primary key value is not syntactically correct, prompt *
  * the user to try again. *

  IF (BAD_SSAN)
    @ 23, 0
    ? CHR(7)
    @ 23, 0 SAY 'SSAN FIELD MUST HAVE NINE (9) DIGITS.  PRESS ANY KEY' ;
              + ' AND TRY AGAIN.'
    WAIT ''
    @ 23, 0
    DONE = .F.
    LOOP
  ENDIF
ENDIF
ENDDO
IF (QUIT_KEY)
  @ 18, 0 CLEAR TO 24,79
  @ 21,33 SAY 'CLOSING FILES'
  @ 24, 0
ENDIF
*
RETURN

```

```

*-----*
*                                     SSAN_CHK                                     *
*-----*
*
* SUMMARY:
*       The SSAN_CHK procedure checks each character of the primary key
*       input for spaces.  If a space is found, a flag is set and the
*       controlling procedure (INPUT_KEY) reads the flag and tells the
*       user to try again.
*
* VARIABLE DECLARATIONS:
*
*       Variable Name      Status      Purpose
*       -----
*       CHK_POS            LOCAL      Used as an incremental counter to test
*                                     each character of the primary key (SSAN).
*
*-----*

```

```

PROCEDURE SSAN_CHK
*
  PARAMETERS SSAN_STR
*
  CHK_POS = 1
  BAD_SSAN = .F.
  DO WHILE (CHK_POS <= 9)
    POS_NUM = SUBSTR(SSAN_STR,CHK_POS,1)
    IF (POS_NUM = ' ')
      BAD_SSAN = .T.
    ENDIF
    CHK_POS = CHK_POS + 1
  ENDDO
*
  RETURN

```

```

*-----*
*                               INIT_DB                               *
*-----*
*
* SUMMARY:
*       The INIT_DB procedure sets up the dBASE III PLUS work area
*       environments for all the required relations, i.e. specifies work
*       area IDs, opens data files, specifies index files and erases &
*       rebuilds indexes as required.
*
* INVOKING PROCEDURES:
*
*       Procedure Name      Location
*       -----
*       ADD_REC             RCIS_P2.PRG
*       EDIT_REC            RCIS_P2.PRG
*       VIEW_REC            RCIS_P2.PRG
*       DEL_REC             RCIS_P2.PRG
*       TRANS_REC           RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*       Variable Name      Status      Purpose
*       -----
*       CHK_POS            PARAMETER    Used as an incremental counter to test
*                                       each character of the primary key (SSAN).
*
*-----*

```

#### PROCEDURE INIT\_DB

\* Initailize the Master file and all its associated index files. \*

```

SELECT 1
USE &M_FILE
IF (RECNO() = 1 .AND. EOF())
  EMPTY_M = .T.
  IF FILE(M_NDX_F)
    ERASE &M_NDX_F
  ENDIF
ELSE
  IF .NOT. FILE(M_NDX_F)
    INDEX ON &M_NDX_STR TO &M_NDX
  ENDIF
  IF (FIRST_TIME)
    DO BLD_NDX WITH M_NDX
    FIRST_TIME = .F.
  ENDIF
  SET INDEX TO &NDX_LIST
ENDIF

```

\* Initailize the Pay file and its associated index file. \*

```

SELECT 2
USE &P_FILE

```

```

IF (RECNO() = 1 .AND. EOF())
  EMPTY_P = .T.
  IF FILE(P_NDX_F)
    ERASE &P_NDX_F
  ENDIF
ELSE
  IF .NOT. FILE(P_NDX_F)
    INDEX ON &P_NDX_STR TO &P_NDX
  ENDIF
  SET INDEX TO &P_NDX
ENDIF

* Initailize the Enrollment totals support file and its associated *
* index files. *

SELECT 3
USE &CT_FILE
IF .NOT. FILE(CT_NDX_F)
  INDEX ON AS_CLASS TO &CT_NDX
ENDIF
SET INDEX TO &CT_NDX
*
RETURN

```

```

*-----*
*                                BLD_NDX                                *
*-----*
*
* SUMMARY:
*   The BLD_NDX procedure checks for the existence of all the index
*   files used to process the queries.  It builds a string of the
*   existing file names to be used whenever the files are updated.
*   These index files must be updated whenever the database files are
*   changed.  If not, the queries will not be able to locate the
*   current information stored on the database files.
*
* INVOKING PROCEDURES:
*
*   Procedure Name      Location
*   -----
*   ADD_REC             RCIS_P2.PRG
*   TRANS_REC           RCIS_P2.PRG
*   INIT_DE             RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   MAS_NDX            PARAMETER   String variable which contains the current
*                                   primary key index for the Master file.
*
*   CGDT_NDX           LOCAL       String variables for index file names.
*   CGDT_NDX_F          "          "
*   CLAS_NDX            "          "
*   CLAS_NDX_F          "          "
*   DCFY_NDX            "          "
*   DCFY_NDX_F          "          "
*   SCHA_NDX            "          "
*   SCHA_NDX_F          "          "
*   SEDT_NDX            "          "
*   SEDT_NDX_F          "          "
*   WPSS_NDX            "          "
*   WPSS_NDX_F          "          "
*
*   PREFIX             LOCAL       Used to store a one letter identifier for
*                                   the source files.
*-----*

```

```

PROCEDURE BLD_NDX
*
  PARAMETER MAS_NDX
*
  PRIVATE WPSS_NDX
  PRIVATE SCHA_NDX
  PRIVATE CLAS_NDX
  PRIVATE DCFY_NDX
  PRIVATE CGDT_NDX
  PRIVATE SEDT_NDX

```

```

PRIVATE WPSS_NDX_F
PRIVATE SCHA_NDX_F
PRIVATE CLAS_NDX_F
PRIVATE DCFY_NDX_F
PRIVATE CGDT_NDX_F
PRIVATE SEDT_NDX_F
PRIVATE PREFIX
*
WPSS_NDX = 'X_WPSS'
SCHA_NDX = 'X_SCHA'
CLAS_NDX = 'X_CLAS'
DCFY_NDX = 'X_DCFY'
CGDT_NDX = 'X_CGDT'
SEDT_NDX = 'X_SEDT'
*
PREFIX      = SUBSTR(MAS_NDX,1,1)
WPSS_NDX     = STUFF(WPSS_NDX,1,1,LTRIM(PREFIX))
SCHA_NDX     = STUFF(SCHA_NDX,1,1,LTRIM(PREFIX))
CLAS_NDX     = STUFF(CLAS_NDX,1,1,LTRIM(PREFIX))
DCFY_NDX     = STUFF(DCFY_NDX,1,1,LTRIM(PREFIX))
CGDT_NDX     = STUFF(CGDT_NDX,1,1,LTRIM(PREFIX))
SEDT_NDX     = STUFF(SEDT_NDX,1,1,LTRIM(PREFIX))
WPSS_NDX_F   = WPSS_NDX + '.NDX'
SCHA_NDX_F   = SCHA_NDX + '.NDX'
CLAS_NDX_F   = CLAS_NDX + '.NDX'
DCFY_NDX_F   = DCFY_NDX + '.NDX'
CGDT_NDX_F   = CGDT_NDX + '.NDX'
SEDT_NDX_F   = SEDT_NDX + '.NDX'
*
NDX_LIST = MAS_NDX
IF FILE(WPSS_NDX_F)
  NDX_LIST = NDX_LIST + ',' + WPSS_NDX
ENDIF
IF FILE(SCHA_NDX_F)
  NDX_LIST = NDX_LIST + ',' + SCHA_NDX
ENDIF
IF FILE(CLAS_NDX_F)
  NDX_LIST = NDX_LIST + ',' + CLAS_NDX
ENDIF
IF FILE(DCFY_NDX_F)
  NDX_LIST = NDX_LIST + ',' + DCFY_NDX
ENDIF
IF FILE(CGDT_NDX_F)
  NDX_LIST = NDX_LIST + ',' + CGDT_NDX
ENDIF
IF FILE(SEDT_NDX_F)
  NDX_LIST = NDX_LIST + ',' + SEDT_NDX
ENDIF
*
RETURN

```

```

*-----*
*                               INIT_SAV                               *
*-----*
*
* SUMMARY:
*       The INIT_SAV procedure simply initializes the SAV_REC variables
*       which are used in the updating and deleting processes for PAY
*       records.
*
* INVOKING PROCEDURES:
*
*       Procedure Name          Location
*       -----
*       EDIT_PAY                RCIS_P2.PRG
*       DEL_PAY                 RCIS_P2.PRG
*
*-----*

```

```

PROCEDURE INIT_SAV

```

```

*
  SAV_REC1  = 0
  SAV_REC2  = 0
  SAV_REC3  = 0
  SAV_REC4  = 0
  SAV_REC5  = 0
  SAV_REC6  = 0
  SAV_REC7  = 0
  SAV_REC8  = 0
  SAV_REC9  = 0
  SAV_REC10 = 0
  SAV_REC11 = 0
  SAV_REC12 = 0
  SAV_REC13 = 0
  SAV_REC14 = 0
  SAV_REC15 = 0
  SAV_REC16 = 0

```

```

*
RETURN

```



```

*-----*
*                               INIT_FLG                               *
*-----*
*
* SUMMARY:                                                              *
*       The INIT_FLG procedure simply initializes the FLAG_REC variables *
*       which are used in the deleting processes for PAY records.      *
*
*-----*

```

# PROCEDURE INIT\_FLG

```

*
FLAG_REC1  = .F.
FLAG_REC2  = .F.
FLAG_REC3  = .F.
FLAG_REC4  = .F.
FLAG_REC5  = .F.
FLAG_REC6  = .F.
FLAG_REC7  = .F.
FLAG_REC8  = .F.
FLAG_REC9  = .F.
FLAG_REC10 = .F.
FLAG_REC11 = .F.
FLAG_REC12 = .F.
FLAG_REC13 = .F.
FLAG_REC14 = .F.
FLAG_REC15 = .F.
FLAG_REC16 = .F.

```

```

*
RETURN

```

```

*-----*
*                               SAV_RECS                               *
*-----*
*
* SUMMARY:
*       The SAV_RECS procedure is used in coordination with the procedures*
*       dures that process all Pay records on the same screen(16 maximum).*
*       It saves the database record numbers which correspond to the line *
*       they are displayed on so that the system knows which screen line *
*       to use in displaying the appropriate records.
*
* INVOKING PROCEDURES:
*
*       Procedure Name           Location
*       -----
*       EDIT_PAY                 RCIS_P2.PRG
*       DEL_PAY                  RCIS_P2.PRG
*
*-----*

```

# PROCEDURE SAV\_RECS

```

*
DO CASE
  CASE DISP_LINE = 1
    SAV_REC1 = RECNO()
  CASE DISP_LINE = 2
    SAV_REC2 = RECNO()
  CASE DISP_LINE = 3
    SAV_REC3 = RECNO()
  CASE DISP_LINE = 4
    SAV_REC4 = RECNO()
  CASE DISP_LINE = 5
    SAV_REC5 = RECNO()
  CASE DISP_LINE = 6
    SAV_REC6 = RECNO()
  CASE DISP_LINE = 7
    SAV_REC7 = RECNO()
  CASE DISP_LINE = 8
    SAV_REC8 = RECNO()
  CASE DISP_LINE = 9
    SAV_REC9 = RECNO()
  CASE DISP_LINE = 10
    SAV_REC10 = RECNO()
  CASE DISP_LINE = 11
    SAV_REC11 = RECNO()
  CASE DISP_LINE = 12
    SAV_REC12 = RECNO()
  CASE DISP_LINE = 13
    SAV_REC13 = RECNO()
  CASE DISP_LINE = 14
    SAV_REC14 = RECNO()
  CASE DISP_LINE = 15
    SAV_REC15 = RECNO()
  CASE DISP_LINE = 16
    SAV_REC16 = RECNO()

```

ENDCASE  
\*  
RETURN

```

*-----*
*                                RCIS_HDR                                *
*-----*
* SUMMARY:                                                                *
*      The RCIS_HDR procedure redisplay the selected mode by repainting*
*      the pop-up menus.                                                *
*-----*
* INVOKING PROCEDURES:
*
*      Procedure Name      Location
*      -----
*      ADD_REC             RCIS_P2.PRG
*      ADD_PAY             RCIS_P2.PRG
*      EDIT_REC            RCIS_P2.PRG
*      EDIT_PAY            RCIS_P2.PRG
*      DEL_REC             RCIS_P2.PRG
*      DEL_PAY             RCIS_P2.PRG
*      VIEW_REC            RCIS_P2.PRG
*      VIEW_PAY            RCIS_P2.PRG
*      TRANS_REC           RCIS_P2.PRG
*-----*

```

# PROCEDURE RCIS\_HDR

```

*
SET ESCAPE OFF
SET SCOREBOARD OFF
SET FILTER TO
SET FORMAT TO
CLEAR GETS
@ 1, 0 TO 3,79
@ 2,22 SAY 'ROTC CADET INFORMATION SYSTEM (RCIS)'
CALL MENU WITH F_PARA
CALL MENU WITH G_PARA
IF (F_SELECT = 'M')
    CALL MENU WITH QS_PARA
    CALL MENU WITH QO_PARA
ELSE
    IF (F_SELECT <> 'L')
        CALL MENU WITH R_PARA
    ENDIF
ENDIF
@ 24, 0
*
RETURN

```

```

*-----*
*                                ERR_RE                                *
*-----*
*                                *
* SUMMARY:                      *
*      The ERR_RE procedure displays an error message informing the user*
*      that a record with the requested key value already exists and    *
*      then accepts a continuation option.                             *
*                                *
* INVOKING PROCEDURES:          *
*                                *
*                                Procedure Name      Location            *
*                                -----            -
*                                ADD_REC              RCIS_P2.PRG          *
*                                *
*-----*

```

# PROCEDURE ERR\_RE

```

*
? CHR(7)
M_CHOICE = .T.
@ 18, 0 CLEAR TO 24,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'

* If the selected function is not Add or Transfer, display the *
* secondary key values for the selected record.                *

IF (F_SELECT <> 'H') .AND. (F_SELECT <> 'L')
  @ 19, 5 SAY 'First Name'
  @ 20, 4 SAY 'Middle Name'
  @ 21, 6 SAY 'Last Name'
  @ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
  @ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
  @ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
ENDIF
@ 23,10 SAY 'RECORD ALREADY EXISTS. DO YOU WANT TO TRY AGAIN [Y/N]? ';
      GET M_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
@ 18, 0 CLEAR TO 24,79
IF .NOT. M_CHOICE
  @ 21,33 SAY 'CLOSING FILES'
ENDIF
*
RETURN

```

```

*-----*
*                                ERR_NF                                *
*-----*
*
* SUMMARY:
* The ERR_NF procedure displays an error message informing the user*
* that a record with the requested key value doesn't exist and then*
* accepts a continuation option.
*
* INVOKING PROCEDURES:
*
* Procedure Name      Location
*-----*
* ADD_REC             RCIS_P2.PRG
* EDIT_REC            RCIS_P2.PRG
* EDIT_PAY            RCIS_P2.PRG
* DEL_REC             RCIS_P2.PRG
* DEL_PAY             RCIS_P2.PRG
* VIEW_REC            RCIS_P2.PRG
* VIEW_PAY            RCIS_P2.PRG
* TRANS_REC           RCIS_P2.PRG
*-----*

```

# PROCEDURE ERR\_NF

```

*
? CHR(7)
M_CHOICE = .T.
@ 18, 0 CLEAR TO 22,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'

* If the selected function is not Add or Transfer, display the *
* secondary key values for the selected record.
*

IF (F_SELECT <> 'H') .AND. (F_SELECT <> 'L')
  @ 19, 5 SAY 'First Name'
  @ 20, 4 SAY 'Middle Name'
  @ 21, 6 SAY 'Last Name'
  @ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
  @ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
  @ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
ENDIF
@ 23,11 CLEAR TO 23,79
@ 23,11 SAY 'RECORD NOT FOUND. DO YOU WANT TO TRY AGAIN [Y/N]? ';
      GET M_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
@ 18, 0 CLEAR TO 24,79
IF .NOT. M_CHOICE
  @ 21,33 SAY 'CLOSING FILES'
ENDIF
*
RETURN

```



```

*-----*
*                                P_PROMPT                                *
*-----*
*                                *
* SUMMARY:                                                                *
*      The P_PROMPT procedure displays a message asking the user if they *
*      would like to add additional Pay records associated with the      *
*      current Master record.                                           *
*-----*

```

```

PROCEDURE P_PROMPT

```

```

*
```

```

*   Display the primary and secondary key values for the selected record.  *
```

```

@ 18, 0 CLEAR TO 24,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'
@ 19, 5 SAY 'First Name'
@ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
@ 20, 4 SAY 'Middle Name'
@ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
@ 21, 6 SAY 'Last Name'
@ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
P_CHOICE = .T.
@ 23, 4 SAY 'WOULD YOU LIKE TO ADD AN ADDITIONAL PAY RECORD'
@ 23,51 SAY 'FOR THIS CADET [Y/N]? ' GET P_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
IF .NOT. P_CHOICE
    @ 18, 0 CLEAR TO 24,79
    @ 21,33 SAY 'CLOSING FILES'
ENDIF
*
RETURN

```



```

*-----*
*                               M_PROMPT                               *
*-----*
*
* SUMMARY:
*       The M_PROMPT procedure displays a continuation message and
*       accepts the user option.
*
* INVOKING PROCEDURES:
*
*       Procedure Name          Location
*       -----
*       ADD_REC                 RCIS_P2.PRG
*       EDIT_REC                RCIS_P2.PRG
*       EDIT_PAY                RCIS_P2.PRG
*       DEL_REC                 RCIS_P2.PRG
*       DEL_PAY                 RCIS_P2.PRG
*       VIEW_REC                RCIS_P2.PRG
*       TRANS_REC               RCIS_P2.PRG
*
*-----*

```

# PROCEDURE M\_PROMPT

```

*
@ 18, 0 CLEAR TO 24,79
M_CHOICE = .T.
@ 23,16 SAY 'DO YOU WANT TO CONTINUE WITH THIS MODE [Y/N]? ';
      GET M_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
IF .NOT. M_CHOICE
  @ 18, 0 CLEAR TO 24,79
  @ 21,33 SAY 'CLOSING FILES'
ENDIF
*
RETURN

```

```

*-----*
*               D_PROMPT               *
*-----*
*
* SUMMARY:
*       The D_PROMPT procedure displays a message requesting confirmation*
*       for record deletion.  The user response is accepted.          *
*
* INVOKING PROCEDURES:
*
*               Procedure Name           Location
*               -----
*               ADD_REC                  RCIS_P2.PRG
*               DEL_REC                  RCIS_P2.PRG
*
*-----*

```

# PROCEDURE D\_PROMPT

```

*
*   Display the primary and secondary key values for the selected record.  *
*
@ 18, 0 CLEAR TO 24,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'
@ 19, 5 SAY 'First Name'
@ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
@ 20, 4 SAY 'Middle Name'
@ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
@ 21, 6 SAY 'Last Name'
@ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
P_CHOICE = .F.
@ 23,20 SAY 'DO YOU WANT TO DELETE THIS RECORD [Y/N]? ';
      GET P_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
*
RETURN

```

```

*-----*
*                                TQ_PRMP*
*-----*
*                                *
* SUMMARY:                      *
*      The TQ_PRMP procedure displays a message requesting confirmation*
*      for record transfer.  The user response is accepted.          *
*-----*

```

# PROCEDURE TQ\_PRMP

\*

\* Display the primary and secondary key values for the selected record. \*

```

@ 18, 0 CLEAR TO 24,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'
@ 19, 5 SAY 'First Name'
@ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
@ 20, 4 SAY 'Middle Name'
@ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
@ 21, 6 SAY 'Last Name'
@ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
TQ_CHOICE = .F.
@ 23,20 SAY 'DO YOU WANT TO TRANSFER THIS RECORD [Y/N]? ';
      GET TQ_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
*
RETURN

```

```

*-----*
*                               VP_PRMPPT                               *
*-----*
*
* SUMMARY:
*       The VP_PRMPPT procedure displays a message asking the user if
*       they would like to view all the Pay records associated with the
*       current Master record.
*
* INVOKING PROCEDURES:
*
*       Procedure Name           Location
*       -----
*       EDIT_PAY                 RCIS_P2.PRG
*       VIEW_PAY                 RCIS_P2.PRG
*
*-----*

```

PROCEDURE VP\_PRMPPT

\*

\* Display the primary and secondary key values for the selected record. \*

```

@ 18, 0 CLEAR TO 24,79
@ 18,11 SAY 'SSAN'
@ 18,16 SAY IN_SSAN PICTURE '@R 999-99-9999'
@ 19, 5 SAY 'First Name'
@ 19,16 SAY IN_FNAM PICTURE '!!!!!!!!!!!!!!'
@ 20, 4 SAY 'Middle Name'
@ 20,16 SAY IN_MNAM PICTURE '!!!!!!!!!!!!!!'
@ 21, 6 SAY 'Last Name'
@ 21,16 SAY IN_LNAM PICTURE '!!!!!!!!!!!!!!'
VP_CHOICE = .F.
@ 23, 4 SAY "DO YOU WANT TO VIEW THIS CADET'S PAY RECORD(S) [Y/N]? " ;
      GET VP_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
*
RETURN

```

```

*-----*
*                                DB3_ERR                                *
*-----*
*
* SUMMARY:
*
*   The DB3_ERR procedure displays system error messages and provides
*   limited corrective action capabilities. If a corrupted index con-
*   dition is detected, the system attempts to repair it by creating a
*   replacement. For other errors, the system will display an advisory
*   message and the error number detected. This error number can be
*   used to locate the problem area. An exact decoding of error num-
*   bers can be found in the dBASE III PLUS User's Manual Appendices.
*
* INVOKING PROCEDURES:
*
*      Procedure Name      Location
*      -----
*      ADD_REC             RCIS_P2.PRG
*      EDIT_REC            RCIS_P2.PRG
*      DEL_REC             RCIS_P2.PRG
*      VIEW_REC            RCIS_P2.PRG
*      TRANS_REC           RCIS_P2.PRG
*
* VARIABLE DECLARATIONS:
*
*      Variable Name      Status      Purpose
*      -----
*      ERR_NUM            PARAMETER    Used to hold the system error number
*                                     returned by the built-in function ERROR().
*
*      ERR_MSG            PARAMETER    Used to hold the system error number re-
*                                     turned by the built-in function MESSAGE().
*
*      PRFX_SAV           LOCAL        Used to store a one letter identifier for
*                                     the source files.
*
*-----*

```

```

PROCEDURE DB3_ERR

```

```

*

```

```

  PARAMETERS ERR_NUM, ERR_MSG

```

```

*

```

```

  PRIVATE PRFX_SAV

```

```

*

```

```

  @ 21, 0
  ? CHR(7)
  @ 21, 0
  ? CHR(7)
  @ 21, 0
  ? CHR(7)

```

```

*   If an index error has occurred, try to correct the error by reindexing
*   all query index files using appropriate index string variables.

```

```

IF (ERR_NUM = 68) .OR. (ERR_NUM = 114)

```

```

@ 21, 0
@ 21,15 SAY 'INDEX ERROR DETECTED.  ATTEMPTING TO REBUILD INDICES.'
@ 24,0
PRFX_SAV = LEFT(NDX_LIST,1)
STR_LEN  = LEN(NDX_LIST)
STRT_POS = 1
DO WHILE (STRT_POS < STR_LEN)
    NDX_NAM  = SUBSTR(NDX_LIST,STRT_POS,6)
    NDX_NAM_F = NDX_NAM + '.NDX'
    NDX_ID   = RIGHT(NDX_NAM,4)
    DO CASE
        CASE NDX_ID = 'SSAN'
            NDX_STR = 'SSAN'
        CASE NDX_ID = 'WPSS'
            NDX_STR = 'AS_CLASS+(WPSS/1000.0)'
        CASE NDX_ID = 'SCHA'
            NDX_STR = 'AS_CLASS+(CUM_GPA/10.0)'
        CASE NDX_ID = 'CLAS'
            NDX_STR = 'STR(AS_CLASS,1)+CAT_TYPE+L_NAME+F_NAME'
        CASE NDX_ID = 'DCFY'
            NDX_STR = 'YEAR(COM_DATE+92)+(FY_RTNG/100.00)';
            + '(DC_RTNG/1000.000)'
        CASE NDX_ID = 'CGDT'
            NDX_STR = 'STR(AS_CLASS,1)+STR(YEAR(COM_DATE),4)';
            + '+STR(MONTH(COM_DATE),2)+STR(DAY(COM_DATE),2)'
        CASE NDX_ID = 'SEDT'
            NDX_STR = 'STR(AS_CLASS,1)+STR(YEAR(SCHLR_DATE),4)';
            + '+STR(MONTH(SCHLR_DATE),2)+STR(DAY(SCHLR_DATE),2)';
            + '+STR(SCHLR_TYPE,3,1)'
    ENDCASE
    IF FILE(NDX_NAM_F)
        REINDEX ON &NDX_STR TO &NDX_NAM
    ENDIF
    STRT_POS = STRT_POS + 7
ENDDO
IF (PRFX_SAV = 'A') .OR. (PRFX_SAV = 'I')
    CL_NDX  = STUFF(CT_NDX,1,1,LTRIM(PRFX_SAV))
    PY_NDX  = STUFF(P_NDX,1,1,LTRIM(PRFX_SAV))
    CL_NDX_F = CL_NDX + '.NDX'
    PY_NDX_F = PY_NDX + '.NDX'
    CL_STR  = 'AS_CLASS'
    PY_STR  = 'SSAN+STR(YEAR(PAY_DATE1),4)+STR(MONTH(PAY_DATE1),2)';
    + '+STR(DAY(PAY_DATE1),2)'
    IF FILE(CL_NDX_F)
        REINDEX ON &CL_STR TO &CL_NDX
    ENDIF
    IF FILE(PY_NDX_F)
        REINDEX ON &PY_STR TO &PY_NDX
    ENDIF
ENDIF
@ 21, 0
? CHR(7)
@ 21,15 SAY 'INDICES REBUILT.  ATTEMPTING TO CONTINUE PROCESSING.'
@ 21, 0
RETRY

```

```

ELSE
  IF (ERR_NUM = 126)
    @ 23, 0
    @ 23,10 SAY 'PRINTER ERROR. CHECK PRINTER AND PRESS ANY KEY TO' ;
              + ' CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ' '
    @ 23, 0
  ELSE
    @ 22, 0
    @ 23, 0
    @ 22, 0 SAY ERR_MSG
    @ 23, 0 SAY 'REPORT ERROR CODE ['
    @ 23,19 SAY ERR_NUM PICTURE '@B ###'
    @ 23,22 SAY ']. PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ' '
    @ 22, 0
    @ 23, 0
  ENDIF
ENDIF
*
RETURN

```

```

*-----*
*               BEGINNING OF RCIS_P3.PRG               *
*-----*
*               QUERIES                                *
*-----*
* SUMMARY:
*     QUERIES is the main driver for the system Query functions.  It
*     prepares the required database files for processing and invokes
*     the specific query procedure that the user has requested.
*
* CALLED PROCEDURES:
*
*               Procedure Name                Location
*               -----
*               SET_DBQ                      RCIS_P3.PRG
*               DB3_Q_ERR                   RCIS_P3.PRG
*               WPSS_QRY                    RCIS_P3.PRG
*               SCHA_QRY                    RCIS_P3.PRG
*               DCFY_QRY                    RCIS_P3.PRG
*               CLAS_QRY                    RCIS_P3.PRG
*               HRAX_QRY                    RCIS_P3.PRG
*               CGDT_QRY                    RCIS_P3.PRG
*               SEDT_QRY                    RCIS_P3.PRG
*               WTAR_QRY                    RCIS_P3.PRG
*               INDV_QRY                    RCIS_P3.PRG
*               PAYI_QRY                    RCIS_P3.PRG
*
* VARIABLE DECLARATIONS:
*
*               Variable Name      Status      Purpose
*               -----
*               QRY_NDX            LOCAL      String variable containing the list of
*               database index file names used by the
*               queries.
*
*               QRY_NDX_F          LOCAL      String variable containing a single data-
*               base index file name.
*
*               PRFX_SAV           LOCAL      Used to save a one letter identifier
*               from the front-end of the index file name
*
*               STRT_POS           LOCAL      Used as a pointer to locate the beginning
*               of each file name in the index string.
*
*-----*

PROCEDURE QUERIES
*
DO SET_DBQ
ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
*
SELECT 1
USE &M_FILE

```



```

IF QS_SELECT = 'Q'
  SELECT 2
  USE &P_FILE
ENDIF
SELECT 1

* If the Master file is empty, erase all existing index files. *

IF (RECNO() = 1 .AND. EOF())
  @ 23, 0 SAY 'REQUIRED DATABASE FILE IS EMPTY.  PRESS ANY KEY AND MAKE' ;
    + ' ANOTHER SELECTION.'
  WAIT ''
  PRFX_SAV = LEFT(M_NDX_F,1)
  QRY_NDX = 'X_SSAN.NDX,X_WPSS.NDX,X_SCHA.NDX,X_DCFY.NDX,X_CLAS.NDX' ;
    + ',X_CGDT.NDX,X_SEDT.NDX'
  STRT_POS = 1
  DO WHILE (STRT_POS < 77)
    QRY_NDX_F = SUBSTR(QRY_NDX,STRT_POS,10)
    QRY_NDX_F = STUFF(QRY_NDX_F,1,1,LTRIM(PREFIX_SAV))
    IF FILE(QRY_NDX_F)
      ERASE &QRY_NDX_F
    ENDIF
    STRT_POS = STRT_POS + 11
  ENDDO
ELSE
  EMPTY_P = .F.
  SELECT 2

  * If the Pay file is empty, erase its index file. *

  IF (RECNO() = 1 .AND. EOF())
    EMPTY_P = .T.
    IF FILE(P_NDX_F)
      ERASE &P_NDX_F
    ENDIF
  ENDIF
  IF (QS_SELECT = 'Q') .AND. (EMPTY_P)
    @ 23, 0 SAY 'REQUIRED DATABASE FILE IS EMPTY.  PRESS ANY KEY AND MAKE' ;
      + ' ANOTHER SELECTION.'
    WAIT ''
  ELSE

    * Initialize spacing variables used in output formatting. *

    S2 = SPACE(2)
    S3 = SPACE(3)
    S4 = SPACE(4)
    S5 = SPACE(5)
    S6 = SPACE(6)
    S7 = SPACE(7)
    S17 = SPACE(17)
    S26 = SPACE(26)
    S31 = SPACE(31)

    * If the WPSS, SCHA, DCFY, or INDV query has been selected, *

```

```

* set up the class enrollment totals relation file.
*

IF (QS_SELECT = 'H' .OR. QS_SELECT = 'I' .OR. QS_SELECT = 'J' .OR. ;
    QS_SELECT = 'P')

*

    SELECT 2
    USE &CT_FILE
    IF (.NOT. FILE(CT_NDX_F))
        INDEX ON AS_CLASS TO &CT_NDX
    ENDIF
    SET INDEX TO &CT_NDX
ENDIF

* If the WTAR query has been selected, set up the height
* standards and the aerobics run time standards relation files. *

IF (QS_SELECT = 'O')
    SELECT 2
    USE T_CDT_HW
    IF (.NOT. FILE('T_HGHT.NDX'))
        INDEX ON HEIGHT TO T_HGHT
    ENDIF
    SET INDEX TO T_HGHT
    SELECT 3
    USE T_CDT_RT
    IF (.NOT. FILE('T_AGEC.NDX'))
        INDEX ON AGE_CAT TO T_AGEC
    ENDIF
    SET INDEX TO T_AGEC
ENDIF

* Direct the process flow to the query procedure which
* corresponds to the user's menu selection.

DO CASE
CASE QS_SELECT = 'H'
    DO WPSS_QRY
CASE QS_SELECT = 'I'
    DO SCHA_QRY
CASE QS_SELECT = 'J'
    DO DCFY_QRY
CASE QS_SELECT = 'K'
    DO CLAS_QRY
CASE QS_SELECT = 'L'
    DO HRAX_QRY
CASE QS_SELECT = 'M'
    DO CGDT_QRY
CASE QS_SELECT = 'N'
    DO SEDT_QRY
CASE QS_SELECT = 'O'
    DO WTAR_QRY
CASE QS_SELECT = 'P'
    DO INDV_QRY
CASE QS_SELECT = 'Q'
    DO PAYI_QRY

```

```

        ENDCASE
    ENDIF
ENDIF
*
SELECT 3
USE
SELECT 2
USE
SELECT 1
USE
*
F_PARA = STUFF(F_PARA,1,1,'A')
F_PARA = STUFF(F_PARA,6,1,'H')
CLEAR
@ 1, 0 TO 3,79
@ 2,22 SAY 'ROTC CADET INFORMATION SYSTEM (RCIS)'
ON ERROR
*
RETURN

```





```

* vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvv *
* Loop until user chooses to terminate this query function mode. *

```

```
DO WHILE (M_CHOICE)
```

```
    * Initialize operator and constraint fields. *
```

```

DONE = .F.
O1A = ' '
F1A = ' '
O1B = ' '
F1B = ' '
O2A = ' '
F2A = ' '
O2B = ' '
F2B = ' '
O3A = ' '
F3A = ' '
O3B = ' '
F3B = ' '
F4 = ' '
PRINT_OPT = 1

```

```

* vvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvv *
* Loop until user enters data in query fields or chooses to terminate *
* this query function mode. *

```

```
DO WHILE (.NOT. DONE)
```

```
    CLEAR
```

```
    DO HELP_SCRN
```

```
    @ 1, 0 TO 16,79
```

```
    @ 1,20 SAY ' WEIGHTED POC SELECTION SYSTEM (WPSS) QUERY '
```

```
    @ 3,28 SAY 'AS Class'
```

```
    @ 6,26 SAY 'WPSS Score'
```

```
    @ 9,27 SAY 'Last Name'
```

```
    @ 12,32 SAY 'SSAN'
```

```
    @ 14,14 SAY 'Print Options'
```

```
    @ 15,14 SAY ' Brief - 1 , Detailed - 2'
```

```

* vvvvvvvvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvvvvvv *
* Loop until user is finished making changes to the input, or until *
* all operator inputs are valid, or until user chooses to terminate *
* this query function mode. *

```

```
DO WHILE (.NOT. DONE)
```

```
    @ 3,37 GET O1A PICTURE '!!!'
```

```
    @ 3,40 GET F1A PICTURE '9'
```

```
    @ 4,37 GET O1B PICTURE '!!!'
```

```
    @ 4,40 GET F1B PICTURE '9'
```

```
    @ 6,37 GET O2A PICTURE '!!!'
```

```
    @ 6,40 GET F2A PICTURE '999'
```

```
    @ 7,37 GET O2B PICTURE '!!!'
```

```
    @ 7,40 GET F2B PICTURE '999'
```

```
    @ 9,37 GET O3A PICTURE '!!!'
```

```

@ 9,40 GET F3A PICTURE '!!!!!!!!!!!!!!'
@ 10,37 GET O3B PICTURE '!!'
@ 10,40 GET F3B PICTURE '!!!!!!!!!!!!!!'
) 12,40 GET F4 PICTURE '@R 999-99-9999'
@ 15,40 GET PRINT_OPT PICTURE '9' RANGE 1,2
CLEAR TYPEAHEAD

```

\* Read query screen inputs and prepare to process them. \*

```

READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

\* If the user chooses to cancel the query, set the required \*  
 \* flags to terminate all procedure loops. \*

```

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE = .F.
  EXIT
ELSE
  STOP_LOOP = .F.
ENDIF

```

\*

```

@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

\* If the user wants to change their inputs, set DONE flag to \*  
 \* false and repeat the current loop. \*

```

IF (DONE)
  @ 23, 0
  DONE = .F.
  LOOP
ELSE
  DONE = .T.
ENDIF

```

\* vvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvv \*  
 \* Check all relational operators for valid entries and exit \*  
 \* the loop when the first invalid entry is detected. \*

```

GOOD_RO = .T.
TEMP_LOOP = .T.
DO WHILE (TEMP_LOOP)
  IF (O1A <> ' ')
    DO RO_CHK WITH O1A
    IF (.NOT. GOOD_RO)
      EXIT

```

```

        ENDIF
    ENDIF
    IF (O1B <> ' ')
        DO RO_CHK WITH O1B
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (O2A <> ' ')
        DO RO_CHK WITH O2A
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (O2B <> ' ')
        DO RO_CHK WITH O2B
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (O3A <> ' ')
        DO RO_CHK WITH O3A
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (O3B <> ' ')
        DO RO_CHK WITH O3B
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
    @ 23, 0
    ? CHR(7)
    M_CHOICE = .F.
    @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR.  WOULD YOU LIKE TO' ;
        + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
    CLEAR TYPEAHEAD
    READ

    * Give the user the option of either returning to the      *
    * query input screen or terminating the query function.    *

    IF (M_CHOICE)
        @ 23, 0
        DONE = .F.
    ELSE
        STOP_LOOP = .T.
        EXIT
    ENDIF
ENDIF
ENDDO

```



```
* Check to see if query termination condition has been previously *
* set to 'true'. *
```

```
IF (STOP_LOOP)
EXIT
```

```
* vvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvv *
* Initialize and build string variables used to define the *
* format for the query output. String variables are used in *
* conjunction with the dBASE III PLUS "SAY" command. *
```

```
ELSE
```

```
HDR1A = ''
HDR1B = ''
HDR2A = ''
HDR2B = ''
HDR3A = ''
HDR3B = ''
DATA1_S = ''
DATA1_L = ''
DATA2_S = ''
DATA2_L = ''
DATA3_S = ''
DATA3_L = ''
SEP_LINE = ''
BLK_LINE = ''
HDR1A = 'First          Last          WPSS      DC      GPA ';
+ ' SAT      AFOQT      AFOQT      AFOQT'
HDR1B = 'Name          Name          Score      Rating      Cum ';
+ ' Cum      AcAp      Quan      Verb '
DATA1_S = "LEFT(F_NAME,14)+S2+L_NAME+S2+STR(WPSS,6,2)+S4";
+ "+STR(DC_RTNG,1)+S5+STR(CUM_GPA,4,2)+S2+STR(SAT_CUM,4)+S2";
+ "+STR(AFOQT_AA,2)+S5+STR(AFOQT_QUAN,2)+S5+STR(AFOQT_VERB,2)+S3"
```

```
*
```

```
IF (PRINT_OPT = 2)
```

```
HDR2A = '          AS      AS Class      GPA ';
+ ' SAT      SAT      Schlr      Pilot'
HDR2B = '          Class      Rank      Sem ';
+ ' Math      Verb      Type      Licns'
DATA2_S = "S31+STR(AS_CLASS,1)+S7+STR(AS_RNK_POS,3)+'/'+CLAS_NUM";
+ "+S3+STR(SEM_GPA,4,2)+S2+STR(SAT_MATH,3)+S3+STR(SAT_VERB,3)";
+ "+S4+TRANSFORM(SCHLR_TYPE,'@ 9.9')+S4+PLS+S4"
HDR3A = '          Phys';
+ '          Grad      Comm      '
HDR3B = '          DOB      Age      Date';
+ '          Date      Date      '
DATA3_S = "S31+DTOC(BIRTHDATE)+S3+AGE+S5+DTOC(PHY_DATE)+S2";
+ "+DTOC(GRAD_DATE)+S2+DTOC(COM_DATE)+S3"
```

```
ENDIF
```

```
SEP_LINE = REPLICATE('-',80)
BLK_LINE = REPLICATE('-',80)
SQG_LINE = REPLICATE('~',80)
```

```
*
```

```
IF (QO_SELECT = 'J')
```

```

HDR1A = HDR1A + ' AFOQT AFOQT Cat FY FSP'
HDR1B = HDR1B + ' Pilot Nav Type Rating Major Date'
DATA1_L = "S2+STR(AFOQT_PLT,2)+S5+STR(AFOQT_NAV,2)+S5";
        + "+CAT_TYPE+S5+STR(FY_RTNG,2)+S6+MAJOR+S3+DTOC(FSP_DATE)"
HDR2A = HDR2A + ' 4-Yr Prior Waiv'
HDR2B = HDR2B + ' Cadet Serv Req Race'
DATA2_L = "S2+FYC+S6+PRS+S6+WRQ+S6+RACE"
HDR3A = HDR3A + ' Form Corps'
HDR3B = HDR3B + ' 48 Auxiliaries'
DATA3_L = "S2+DTOC(FORM_48)+S2" ;
        + "+TRANSFORM(CORPS_AUX,'@R !!!!!!!!!!!!!!!!!!!!!!!')";
SEP_LINE = SEP_LINE + REPLICATE('-',52)
BLK_LINE = BLK_LINE + REPLICATE('-',52)
SQG_LINE = SQG_LINE + REPLICATE('-',52)
ENDIF

```

```

* vvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvv *
* Initialize and build string variable used to set the filter *
* condition for this query. The string variable is used in *
* conjunction with the dBASE III PLUS command "SET FILTER TO". *
* The filter masks all records which do not meet all the con- *
* ditions specified in the string variable. *

```

```

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
    FILT_STR = 'AS_CLASS' + O1A + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
    ELSE
        FILT_STR = 'AS_CLASS' + O1B + F1B
    ENDIF
ENDIF
IF (LEN(LTRIM(F2A)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.WPSS' + O2A + F2A
    ELSE
        FILT_STR = 'WPSS' + O2A + F2A
    ENDIF
ENDIF
IF (LEN(LTRIM(F2B)) > 0 .AND. (O2A <> O2B) .AND. (F2A <> F2B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.WPSS' + O2B + F2B
    ELSE
        FILT_STR = 'WPSS' + O2B + F2B
    ENDIF
ENDIF
IF (LEN(LTRIM(F3A)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.L_NAME' + O3A + "'" + F3A + "'"
    ELSE
        FILT_STR = 'L_NAME' + O3A + "'" + F3A + "'"
    ENDIF
ENDIF
ENDIF

```

```

IF (LEN(LTRIM(F3B)) > 0 .AND. (O3A <> O3B) .AND. (F3A <> F3B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O3B + ' ' + F3B + ' '
  ELSE
    FILT_STR = 'L_NAME' + O3B + ' ' + F3B + ' '
  ENDIF
ENDIF
IF (LEN(LTRIM(F4)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.SSAN =' + ' ' + F4 + ' '
  ELSE
    FILT_STR = 'SSAN =' + ' ' + F4 + ' '
  ENDIF
ENDIF
DONE = .T.

```

```

* vvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvv *
* If user has entered data in the query fields, then proceed to *
* process their inputs. Open the required database files, set *
* the filter condition, set the print constraints and direct the*
* print to the selected output media. *

```

```

IF (LEN(FILT_STR) > 0)
  @ 23, 0
  @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
  SELECT 1
  IF (.NOT. FILE(M_NDX_F))
    INDEX ON &M_NDX_STR TO &M_NDX
  ENDIF
  SET INDEX TO &M_NDX
  SET FILTER TO &FILT_STR
  GOTO TOP
DO CASE

```

```

  * If none of the database records meet all the input *
  * constraints, give the user the option to try again *
  * or to terminate the query. *

```

```

CASE (EOF())
  DO ERR_NF
  IF (M_CHOICE)
    DONE = .F.
    LOOP
  ELSE
    EXIT
  ENDIF

```

```

* If some database records meet the constraints, ini- *
* tialize the print environment and perform print loop *
* until all records are printed. *

```

```

CASE (.NOT. EOF())
  IF QO_SELECT <> 'H'
    SET PRINT ON
    SET DEVICE TO PRINT
  
```

```

        IF QO_SELECT = 'J'
            @ 0, 1 SAY CHR(27) + CHR(15)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(77)
        ENDIF
        MAX_LINES = 66
    ELSE
        MAX_LINES = 23
    ENDIF
    IF (QO_SELECT <> 'J')
        SPACER = SPACE(18)
    ELSE
        SPACER = SPACE(49)
    ENDIF
    CLEAR
    @ 0, 0 SAY SPACER + 'WEIGHTED POC SELECTION SYSTEM';
        + '(WPSS) REPORT'
    @ 1, 0
    FIRST_TIME = .T.
    DISP_LINE = 2

* vvvvvvvvvvvv #8. DATABASE RECORD LOOP vvvvvvvvvvvv *
* Loop until all database records (which meet input *
* constraints) have been printed. *

DO WHILE (.NOT. EOF())
    IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
        IF (.NOT. FIRST_TIME)
            EJECT
        ENDIF
    ENDIF
    IF (FIRST_TIME)
        FIRST_TIME = .F.
    ELSE
        DISP_LINE = 0
        CLEAR
    ENDIF

* vvvvvvvvvvvvvvvv #9. PAGING LOOP vvvvvvvvvvvvvvvv *
* Loop until the display line exceeds the maximum *
* number of lines for the selected output media. *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))
    REC_NUM = RECNO()

* If the number of print lines per cadet will *
* not fit on one page, exit the loop and go to *
* the next page. *

    IF ((MAX_LINES-DISP_LINE) < 11).AND.(PRINT_OPT = 2)
        EXIT
    ELSE
        IF ((DISP_LINE <= 3) .OR. (PRINT_OPT = 2))
            @ DISP_LINE, 0 SAY HDR1A
            @ DISP_LINE + 1, 0 SAY HDR1B
        
```

```

        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 1, 0 SAY SEP_LINE
        ENDIF
        DISP_LINE = DISP_LINE + 2
    ENDIF
    @ DISP_LINE, 0 SAY &DATA1_S
    IF (QO_SELECT = 'J')
        @ DISP_LINE, 80 SAY &DATA1_L
    ENDIF
    DISP_LINE = DISP_LINE + 2
    IF (PRINT_OPT = 2)
        @ DISP_LINE, 0 SAY HDR2A
        @ DISP_LINE + 1, 0 SAY HDR2B
        IF (QO_SELECT <> 'H')
            SEP_LINE = STUFF(SEP_LINE,1,31,S31)
            @ DISP_LINE + 1, 0 SAY SEP_LINE
        ENDIF
        PLS = 'N'
        IF PLT_LICENS
            PLS = 'Y'
        ENDIF
        CLAS_VAL = AS_CLASS
        SELECT 2
        SEEK CLAS_VAL
        IF (.NOT. EOF())
            CLAS_NUM = STR(AS_CL_TOT,3)
        ELSE
            CLAS_NUM = ' ? '
        ENDIF
        SELECT 1
        GOTO REC_NUM
        @ DISP_LINE + 2, 0 SAY &DATA2_S
        IF (QO_SELECT = 'J')
            FYC = 'N'
            PRS = 'N'
            WRQ = 'N'
            IF FOUR_YR
                FYC = 'Y'
            ENDIF
            IF PRIOR_SVC
                PRS = 'Y'
            ENDIF
            IF WAIVER_REQ
                WRQ = 'Y'
            ENDIF
            @ DISP_LINE + 2, 80 SAY &DATA2_L
        ENDIF
        @ DISP_LINE + 4, 0 SAY HDR3A
        @ DISP_LINE + 5, 0 SAY HDR3B
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 5, 0 SAY SEP_LINE
            SEP_LINE = STUFF(SEP_LINE,1,31,REPLICATE('_',31))
        ENDIF
        @ DISP_LINE + 6, 0 SAY &DATA3_S
        IF (QO_SELECT = 'J')

```

```

        @ DISP_LINE + 6, 80 SAY &DATA3_L
    ENDIF
    @ DISP_LINE + 7, 0 SAY SQG_LINE
    DISP_LINE = DISP_LINE + 8
ENDIF
ENDIF

* Issue dBASE III PLUS command to go to the      *
* next record which meets the input constraints.*

SKIP
ENDDO

* If the output media is the screen, issue the user*
* paging prompt.                                  *

IF (QO_SELECT = 'H')
    @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
    CLEAR TYPEAHEAD
    WAIT ''
ENDIF
ENDDO
IF (QO_SELECT <> 'H')
    @ DISP_LINE + 1, 0 SAY CHR(10)
    EJECT
    IF (QO_SELECT = 'J')
        @ 0, 1 SAY CHR(18)
    ELSE
        @ 0, 1 SAY CHR(27) + CHR(80)
    ENDIF
    SET PRINT OFF
ENDIF
SET DEVICE TO SCREEN
SET FILTER TO

ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure).                                *

ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 4 SAY 'PLEASE ENTER DATA.  PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ''
    @ 23, 0
    DONE = .F.
ENDIF
ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *

```

```

* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
    DO RCIS_HDR
    DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 2
USE
SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                               SCHA_QRY                               *
*-----*
*                               *
* SUMMARY:                               *
*   The SCHA_QRY procedure provides the interface for the user to per- *
*   form ad hoc queries on cadet data which is related to cadet schol- *
*   arship requirements and/or cadet academic performance.           *
*                               *
*-----*

```

PROCEDURE SCHA\_QRY

\*

PRIVATE SPACER

\*

ON ERROR DO DB3\_Q\_ERR WITH ERROR(), MESSAGE()

CLEAR

M\_CHOICE = .T.

\* vvvvvvvvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvvvvvvvvvv \*

DO WHILE (M\_CHOICE)

\* Initialize operator and constraint fields. \*

DONE = .F.

O1A = ' '

F1A = ' '

O1B = ' '

F1B = ' '

F2 = ' '

O3A = ' '

F3A = ' '

O3B = ' '

F3B = ' '

O4 = '>='

F4 = ' '

O5 = '>='

F5 = '10'

O6 = '>='

F6 = '15'

O7 = '>='

F7 = '50'

O8 = '>='

F8 = '30'

O9 = ' '

F9 = ' '

\* vvvvvvvvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvvvvvvvvvv \*

DO WHILE (.NOT. DONE)

CLEAR

DO HELP\_SCRN

@ 1, 0 TO 15,79



```

@ 1,16 SAY ' SCHOLARSHIP CANDIDATES/ACADEMIC PERFORMANCE QUERY '
@ 3,12 SAY 'AS Class'
@ 6, 2 SAY 'Scholarship'
@ 7, 2 SAY 'Category (T, N, P)'
@ 10,11 SAY 'Last Name'
@ 3,49 SAY 'Cumulative GPA'
@ 5,53 SAY 'AFOQT Quan'
@ 7,53 SAY 'AFOQT Verb'
@ 9,52 SAY 'AFOQT Pilot'
@ 11,54 SAY 'AFOQT Nav'
@ 13,49 SAY 'Cumulative SAT'

```

```

* vvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvv *

```

```

DO WHILE (.NOT. DONE)
@ 3,21 GET O1A PICTURE '!!'
@ 3,24 GET F1A PICTURE '9'
@ 4,21 GET O1B PICTURE '!!'
@ 4,24 GET F1B PICTURE '9'
@ 7,24 GET F2 PICTURE '!'
@ 10,21 GET O3A PICTURE '!!'
@ 10,24 GET F3A PICTURE '!!!!!!!!!!!!!!!!!!!!'
@ 11,21 GET O3B PICTURE '!!'
@ 11,24 GET F3B PICTURE '!!!!!!!!!!!!!!!!!!!!'
@ 3,64 GET O4 PICTURE '!!'
@ 3,67 GET F4 PICTURE '9.99'
@ 5,64 GET O5 PICTURE '!!'
@ 5,67 GET F5 PICTURE '99'
@ 7,64 GET O6 PICTURE '!!'
@ 7,67 GET F6 PICTURE '99'
@ 9,64 GET O7 PICTURE '!!'
@ 9,67 GET F7 PICTURE '99'
@ 11,64 GET O8 PICTURE '!!'
@ 11,67 GET F8 PICTURE '99'
@ 13,64 GET O9 PICTURE '!!'
@ 13,67 GET F9 PICTURE '9999'
READ

```

```

* Read query screen inputs and prepare to process them. *

```

```

@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

```

* If the user chooses to cancel the query, set the required *
* flags to terminate all procedure loops. *

```

```

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE = .F.
  EXIT
ELSE
  STOP_LOOP = .F.

```

ENDIF

\*

@ 23, 0

@ 23,19 SAY ;

"DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'

CLEAR TYPEAHEAD

READ

\* If the user wants to change their inputs, set DONE flag to \*  
\* false and repeat the current loop. \*

IF (DONE)

@ 23, 0

DONE = .F.

LOOP

ELSE

DONE = .T.

ENDIF

\* vvvvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvv \*

GOOD\_RO = .T.

TEMP\_LOOP = .T.

DO WHILE (TEMP\_LOOP)

IF (O1A <> ' ')

DO RO\_CHK WITH O1A

IF (.NOT. GOOD\_RO)

EXIT

ENDIF

ENDIF

IF (O1B <> ' ')

DO RO\_CHK WITH O1B

IF (.NOT. GOOD\_RO)

EXIT

ENDIF

ENDIF

IF (O3A <> ' ')

DO RO\_CHK WITH O3A

IF (.NOT. GOOD\_RO)

EXIT

ENDIF

ENDIF

IF (O3B <> ' ')

DO RO\_CHK WITH O3B

IF (.NOT. GOOD\_RO)

EXIT

ENDIF

ENDIF

IF (O4 <> ' ')

DO RO\_CHK WITH O4

IF (.NOT. GOOD\_RO)

EXIT

ENDIF

ENDIF

IF (O5 <> ' ')

```

        DO RO_CHK WITH 05
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (06 <> ' ')
        DO RO_CHK WITH 06
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (07 <> ' ')
        DO RO_CHK WITH 07
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (08 <> ' ')
        DO RO_CHK WITH 08
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (09 <> ' ')
        DO RO_CHK WITH 09
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
    @ 23, 0
    ? CHR(7)
    M_CHOICE = .F.
    @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR.  WOULD YOU LIKE TO' ;
        + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
    CLEAR TYPEAHEAD
    READ

    * Give the user the option of either returning to the      *
    * query input screen or terminating the query function.    *

    IF (M_CHOICE)
        @ 23, 0
        DONE = .F.
    ELSE
        STOP_LOOP = .T.
        EXIT
    ENDIF
ENDIF
IF ((F2 <> 'T').AND.(F2 <> 'N').AND.(F2 <> 'P').AND.(F2 <> ' '))
    @ 23, 0
    ? CHR(7)
    M_CHOICE = .F.

```

```

@ 23, 4 SAY 'INVALID SCHOLARSHIP CATEGORY. WOULD YOU LIKE TO' ;
      + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

* Give the user the option of returning to correct their *
* invalid entry or to terminate the query function.      *

IF (M_CHOICE)
  @ 23, 0
  DONE = .F.
ELSE
  STOP_LOOP = .T.
  EXIT
ENDIF
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'.                                              *

IF (STOP_LOOP)
  EXIT
ELSE

* vvvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvvv *

HDR1A = ''
HDR1B = ''
DATA1_S = ''
DATA1_L = ''
HDR1A = 'First           Last           AS   Cat   GPA   SAT ' ;
      + ' AFOQT
HDR1B = 'Name           Name           Class Type Cum   Cum ' ;
      + ' Quan   Verb   Plt   Nav'
DATA1_S = "F_NAME+S2+L_NAME+S2+STR(AS_CLASS,1)+S6+CAT_TYPE+S5";
+ "+STR(CUM_GPA,4,2)+S2+STR(SAT_CUM,4)+S2+STR(AFOQT_QUAN,2)+S5";
+ "+STR(AFOQT_VERB,2)+S4+STR(AFOQT_PLT,2)+S3+STR(AFOQT_NAV,2)+' '"
SEP_LINE = REPLICATE('-',80)

*

IF (QO_SELECT = 'J')
  HDR1A = HDR1A + '           AFOQT   ACT   WPSS   AS Class   FY ' ;
      + ' GPA'
  HDR1B = HDR1B + '   AcAp   Date           Cum   Score           Rank   Rating' ;
      + ' Sem'
  DATA1_L = "S2+STR(AFOQT_AA,2)+S4+DTC(AFOQT_DATE)+S2";
      + "+STR(ACT_CUM,2)+S3+STR(WPSS,6,2)+S3+STR(AS_RNK_POS,3)+' / '";
      + "+CLAS_NUM+S4+STR(FY_RTNG,2)+S4+STR(SEM_GPA,4,2)"
  SEP_LINE = SEP_LINE + REPLICATE('-',52)
ENDIF

* vvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvv *

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)

```

```

      FILT_STR = 'AS_CLASS' + O1A + F1A
    ENDIF
    IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
      IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
      ELSE
        FILT_STR = 'AS_CLASS' + O1B + F1B
      ENDIF
    ENDIF
    IF (LEN(LTRIM(F2)) > 0)
      IF (LEN(FILT_STR) > 0)
        IF (LTRIM(F2) = 'T')
          FILT_STR = FILT_STR + ".AND.(CAT_TYPE='N'.OR.CAT_TYPE='M';
            + ".OR.CAT_TYPE='2')\""
        ELSE
          IF (LTRIM(F2) = 'N')
            FILT_STR = FILT_STR + ".AND.CAT_TYPE='3'"
          ELSE
            FILT_STR = FILT_STR + ".AND.CAT_TYPE='P'"
          ENDIF
        ENDIF
      ELSE
        IF (LTRIM(F2) = 'T')
          FILT_STR = "(CAT_TYPE='N'.OR.CAT_TYPE='M'.OR.CAT_TYPE='2')\""
        ELSE
          IF (LTRIM(F2) = 'N')
            FILT_STR = "CAT_TYPE='3'"
          ELSE
            FILT_STR = "CAT_TYPE='P'"
          ENDIF
        ENDIF
      ENDIF
    ENDIF
  ENDIF
ENDIF
IF (LEN(LTRIM(F3A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O3A + "'" + F3A + "'"
  ELSE
    FILT_STR = 'L_NAME' + O3A + "'" + F3A + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F3B)) > 0 .AND. (O3A <> O3B) .AND. (F3A <> F3B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O3B + "'" + F3B + "'"
  ELSE
    FILT_STR = 'L_NAME' + O3B + "'" + F3B + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(TRIM(F4))) > 0) .AND. (LTRIM(TRIM(F4)) <> '.')
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.CUM_GPA' + O4 + F4
  ELSE
    FILT_STR = 'CUM_GPA' + O4 + F4
  ENDIF
ENDIF
IF (LEN(LTRIM(F5)) > 0)

```

```

        IF (LEN(FILT_STR) > 0)
            FILT_STR = FILT_STR + '.AND.AFOQT_QUAN' + 05 + F5
        ELSE
            FILT_STR = 'AFOQT_QUAN' + 05 + F5
        ENDIF
    ENDIF
    IF (LEN(LTRIM(F6)) > 0)
        IF (LEN(FILT_STR) > 0)
            FILT_STR = FILT_STR + '.AND.AFOQT_VERB' + 06 + F6
        ELSE
            FILT_STR = 'AFOQT_VERB' + 06 + F6
        ENDIF
    ENDIF
    IF (LEN(LTRIM(F7)) > 0)
        IF (LEN(FILT_STR) > 0)
            FILT_STR = FILT_STR + '.AND.AFOQT_PLT' + 07 + F7
        ELSE
            FILT_STR = 'AFOQT_PLT' + 07 + F7
        ENDIF
    ENDIF
    IF (LEN(LTRIM(F8)) > 0)
        IF (LEN(FILT_STR) > 0)
            FILT_STR = FILT_STR + '.AND.AFOQT_NAV' + 08 + F8
        ELSE
            FILT_STR = 'AFOQT_NAV' + 08 + F8
        ENDIF
    ENDIF
    IF (LEN(LTRIM(F9)) > 0)
        IF (LEN(FILT_STR) > 0)
            FILT_STR = FILT_STR + '.AND.SAT_CUM' + 09 + F9
        ELSE
            FILT_STR = 'SAT_CUM' + 09 + F9
        ENDIF
    ENDIF
    DONE = .T.

*  vvvvvvvvvvv #7.  ACCESS DATABASE & DIRECT OUTPUT  vvvvvvvvvvv *

IF (LEN(FILT_STR) > 0)
    @ 23, 0
    @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
    SELECT 1
    IF (.NOT. FILE(M_NDX_F))
        INDEX ON &M_NDX_STR TO &M_NDX
    ENDIF
    SET INDEX TO &M_NDX
    SET FILTER TO &FILT_STR
    GOTO TOP
DO CASE

    *  If none of the database records meet all the input  *
    *  constraints, give the user the option to try again  *
    *  or to terminate the query.                          *

    CASE (EOF())

```

```

DO ERR_NF
IF (M_CHOICE)
    DONE = .F.
    LOOP
ELSE
    EXIT
ENDIF

* If some database records meet the constraints, ini- *
* tialize the print environment and perform print loop *
* until all records are printed. *

CASE (.NOT. EOF())
    IF QO_SELECT <> 'H'
        SET PRINT ON
        SET DEVICE TO PRINT
        IF QO_SELECT = 'J'
            @ 0, 1 SAY CHR(27) + CHR(15)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(77)
        ENDIF
        MAX_LINES = 66
    ELSE
        MAX_LINES = 23
    ENDIF
    IF (QO_SELECT <> 'J')
        SPACER = SPACE(15)
    ELSE
        SPACER = SPACE(46)
    ENDIF
    CLEAR
    @ 0, 0 SAY SPACER + 'SCHOLARSHIP CANDIDATES/ACADEMIC';
        + ' PERFORMANCE REPORT'
    @ 1, 0
    FIRST_TIME = .T.
    DISP_LINE = 2

* vvvvvvvvvvv #8. DATABASE RECORD LOOP vvvvvvvvvvv *

DO WHILE (.NOT. EOF())
    IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
        IF (.NOT. FIRST_TIME)
            EJECT
        ENDIF
    ENDIF
    IF (FIRST_TIME)
        FIRST_TIME = .F.
    ELSE
        DISP_LINE = 0
        CLEAR
    ENDIF

* vvvvvvvvvvvvvvv #9. PAGING LOOP vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))

```

```

REC_NUM = RECNO()
IF (DISP_LINE <= 3)
  @ DISP_LINE, 0 SAY HDR1A
  @ DISP_LINE + 1, 0 SAY HDR1B
  IF (QO_SELECT <> 'H')
    @ DISP_LINE + 1, 0 SAY SEP_LINE
  ENDIF
  DISP_LINE = DISP_LINE + 2
ENDIF
@ DISP_LINE, 0 SAY &DATA1_S
IF (QO_SELECT = 'J')
  CLAS_VAL = AS_CLASS
  SELECT 2
  SEEK CLAS_VAL
  IF (.NOT. EOF())
    CLAS_NUM = STR(AS_CL_TOT,3)
  ELSE
    CLAS_NUM = ' ? '
  ENDIF
  SELECT 1
  GOTO REC_NUM
  @ DISP_LINE, 80 SAY &DATA1_L
ENDIF
DISP_LINE = DISP_LINE + 2

* Issue dBASE III PLUS command to go to the      *
* next record which meets the input constraints.*

SKIP
ENDDO

* If the output media is the screen, issue the user*
* paging prompt.                                     *

IF (QO_SELECT = 'H')
  @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
  CLEAR TYPEAHEAD
  WAIT ''
ENDIF
ENDDO
IF (QO_SELECT <> 'H')
  @ DISP_LINE + 1, 0 SAY CHR(10)
  EJECT
  IF (QO_SELECT = 'J')
    @ 0, 1 SAY CHR(18)
  ELSE
    @ 0, 1 SAY CHR(27) + CHR(80)
  ENDIF
  SET PRINT OFF
ENDIF
SET DEVICE TO SCREEN
SET FILTER TO
ENDCASE

* If the user fails to enter any data in the input fields, *
```



```

* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure). *

ELSE
  @ 23, 0
  ? CHR(7)
  @ 23, 4 SAY 'PLEASE ENTER DATA.  PRESS ANY KEY TO CONTINUE.'
  CLEAR TYPEAHEAD
  WAIT ''
  @ 23, 0
  DONE = .F.
ENDIF
ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *
* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
  DO RCIS_HDR
  DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 2
USE
SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                               DCFY_QRY                               *
*-----*
*
* SUMMARY:
*   The DCFY_QRY procedure provides the interface for the user to per-
*   form ad hoc queries on cadet data which is related to specified
*   cadet ratings for all cadets being commissioned within a given
*   fiscal year or range of fiscal years.
*-----*

```

# ``` PROCEDURE DCFY_QRY ```

```

*
  PRIVATE SPACER
  PRIVATE FTC
*
  ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
  CLEAR
  M_CHOICE = .T.

  * vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvv *

  DO WHILE (M_CHOICE)

    * Initialize operator and constraint fields. *

    DONE = .F.
    O1A = ' '
    F1A = ' '
    O1B = ' '
    F1B = ' '
    O2A = ' '
    F2A = ' '
    O2B = ' '
    F2B = ' '
    F3 = ' '
    O4A = ' '
    F4A = ' '
    O4B = ' '
    F4B = ' '
    O5A = ' '
    F5A = ' '
    O5B = ' '
    F5B = ' '

    * vvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvv *

    DO WHILE (.NOT. DONE)
      CLEAR
      DO HELP_SCRN
      @ 5, 0 TO 15,79
      @ 5,17 SAY ' DATE OF COMMISSIONING (DOC) FISCAL YEAR QUERY '
      @ 7,11 SAY 'DOC'

```

```

@ 8,11 SAY 'Fiscal Year'
@ 10,13 SAY 'Last Name'
@ 13,18 SAY 'SSAN'
@ 7,52 SAY 'Fiscal Year'
@ 8,52 SAY 'Rating'
@ 10,50 SAY 'Det Commander'
@ 11,50 SAY 'Rating'

```

```

* vvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvv *

```

```

DO WHILE (.NOT. DONE)
  @ 7,23 GET O1A PICTURE '!!'
  @ 7,26 GET F1A PICTURE '99'
  @ 8,23 GET O1B PICTURE '!!'
  @ 8,26 GET F1B PICTURE '99'
  @ 10,23 GET O2A PICTURE '!!'
  @ 10,26 GET F2A PICTURE '!!!!!!!!!!!!!!!!!!'
  @ 11,23 GET O2B PICTURE '!!'
  @ 11,26 GET F2B PICTURE '!!!!!!!!!!!!!!!!!!'
  @ 13,26 GET F3 PICTURE '@R 999-99-9999'
  @ 7,64 GET O4A PICTURE '!!'
  @ 7,67 GET F4A PICTURE '99'
  @ 8,64 GET O4B PICTURE '!!'
  @ 8,67 GET F4B PICTURE '99'
  @ 10,64 GET O5A PICTURE '!!'
  @ 10,67 GET F5A PICTURE '9'
  @ 11,64 GET O5B PICTURE '!!'
  @ 11,67 GET F5B PICTURE '9'

```

```

* Read query screen inputs and prepare to process them. *

```

```

READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

```

* If the user chooses to cancel the query, set the required *
* flags to terminate all procedure loops. *

```

```

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE = .F.
  EXIT
ELSE
  STOP_LOOP = .F.
ENDIF

```

```

*
```

```

@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

```
* If the user wants to change their inputs, set DONE flag to *  
* false and repeat the current loop. *
```

```
IF (DONE)  
  @ 23, 0  
  DONE = .F.  
  LOOP  
ELSE  
  DONE = .T.  
ENDIF
```

```
* vvvvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvv *
```

```
GOOD_RO = .T.  
TEMP_LOOP = .T.  
DO WHILE (TEMP_LOOP)  
  IF (O1A <> ' ')  
    DO RO_CHK WITH O1A  
    IF (.NOT. GOOD_RO)  
      EXIT  
    ENDIF  
  ENDIF  
  IF (O1B <> ' ')  
    DO RO_CHK WITH O1B  
    IF (.NOT. GOOD_RO)  
      EXIT  
    ENDIF  
  ENDIF  
  IF (O2A <> ' ')  
    DO RO_CHK WITH O2A  
    IF (.NOT. GOOD_RO)  
      EXIT  
    ENDIF  
  ENDIF  
  IF (O2B <> ' ')  
    DO RO_CHK WITH O2B  
    IF (.NOT. GOOD_RO)  
      EXIT  
    ENDIF  
  ENDIF  
  IF (O4A <> ' ')  
    DO RO_CHK WITH O4A  
    IF (.NOT. GOOD_RO)  
      EXIT  
    ENDIF  
  ENDIF  
  IF (O4B <> ' ')  
    DO RO_CHK WITH O4B  
    IF (.NOT. GOOD_RO)  
      EXIT  
    ENDIF  
  ENDIF  
  IF (O5A <> ' ')  
    DO RO_CHK WITH O5A  
    IF (.NOT. GOOD_RO)
```

```

        EXIT
    ENDIF
ENDIF
IF (O5B <> ' ')
    DO RO_CHK WITH O5B
    IF (.NOT. GOOD_RO)
        EXIT
    ENDIF
ENDIF
TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
    @ 23, 0
    ? CHR(7)
    M_CHOICE = .F.
    @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR. WOULD YOU LIKE TO' ;
        + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
    CLEAR TYPEAHEAD
    READ

    * Give the user the option of either returning to the      *
    * query input screen or terminating the query function.    *

    IF (M_CHOICE)
        @ 23, 0
        DONE = .F.
    ELSE
        STOP_LOOP = .T.
        EXIT
    ENDIF
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'.                                                  *

IF (STOP_LOOP)
    EXIT
ELSE

* vvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvv *

HDR1A = ''
HDR1B = ''
DATA1_S = ''
DATA1_L = ''
HDR1A = 'First          Last          FY      DC      AS Class';
+ ' AS      Comm      '
HDR1B = 'Name          Name          Rating Rating      Rank  ';
+ ' Class      Date      '
DATA1_S = "F_NAME+S2+L_NAME+S4+STR(FY_RTNG,2)+S6+STR(DC_RTNG,1)+S5";
+ "+STR(AS_RNK_POS,3)+'/'+CLAS_NUM+S5+STR(AS_CLASS,1)+S5";
+ "+DTC( COM_DATE)+S3"
SEP_LINE = REPLICATE('-',80)

```

\*

```

IF (QO_SELECT = 'J')
  HDR1A = HDR1A + 'Grad      Cat   WPSS      GPA      SAT      FT ' ;
      + '      FT'
  HDR1B = HDR1B + 'Date      Type   Score     Cum      Cum      Comp' ;
      + '      Rating'
  DATA1_L = "DTC(GRAD_DATE)+S3+CAT_TYPE+S4+STR(WPSS,6,2)+S3";
      + "+STR(CUM_GPA,4,2)+S3+STR(SAT_CUM,4)+S4";
      + "+FTC+S3+STR(FT_RTNG,6,2)"
  SEP_LINE = SEP_LINE + REPLICATE('_',52)
ENDIF

* vvvvvvvvvvvvvvvvv #6.  BUILD FILTER STRING vvvvvvvvvvvvvvvvv *

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
  FILT_STR = 'YEAR(COM_DATE+92)' + O1A + '19' + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.YEAR(COM_DATE+92)' + O1B + '19' + F1B
  ELSE
    FILT_STR = 'YEAR(COM_DATE+92)' + O1B + '19' + F1B
  ENDIF
ENDIF
IF (LEN(LTRIM(F2A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O2A + "'" + F2A + "'"
  ELSE
    FILT_STR = 'L_NAME' + O2A + "'" + F2A + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F2B)) > 0 .AND. (O2A <> O2B) .AND. (F2A <> F2B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O2B + "'" + F2B + "'"
  ELSE
    FILT_STR = 'L_NAME' + O2B + "'" + F2B + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F3)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.SSAN =' + "'" + F3 + "'"
  ELSE
    FILT_STR = 'SSAN =' + "'" + F3 + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F4A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.FY_RTNG' + O4A + F4A
  ELSE
    FILT_STR = 'FY_RTNG' + O4A + F4A
  ENDIF
ENDIF
IF (LEN(LTRIM(F4B)) > 0 .AND. (O4A <> O4B) .AND. (F4A <> F4B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.FY_RTNG' + O4B + F4B
  ELSE
    FILT_STR = 'FY_RTNG' + O4B + F4B
  ENDIF
ENDIF

```

```

ELSE
    FILT_STR = 'FY_RTNG' + O4B + F4B
ENDIF
ENDIF
IF (LEN(LTRIM(F5A)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.DC_RTNG' + O5A + F5A
    ELSE
        FILT_STR = 'DC_RTNG' + O5A + F5A
    ENDIF
ENDIF
IF (LEN(LTRIM(F5B)) > 0 .AND. (O5A <> O5B) .AND. (F5A <> F5B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.DC_RTNG' + O5B + F5B
    ELSE
        FILT_STR = 'DC_RTNG' + O5B + F5B
    ENDIF
ENDIF
DONE = .T.

* vvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvv *

IF (LEN(FILT_STR) > 0)
    @ 23, 0
    @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
    SELECT 1
    IF (.NOT. FILE(M_NDX_F))
        INDEX ON &M_NDX_STR TO &M_NDX
    ENDIF
    SET INDEX TO &M_NDX
    SET FILTER TO &FILT_STR
    GOTO TOP
    DO CASE

        * If none of the database records meet all the input *
        * constraints, give the user the option to try again *
        * or to terminate the query. *

        CASE (EOF())
            DO ERR_NF
            IF (M_CHOICE)
                DONE = .F.
                LOOP
            ELSE
                EXIT
            ENDIF

        * If some database records meet the constraints, ini- *
        * tialize the print environment and perform print loop *
        * until all records are printed. *

        CASE (.NOT. EOF())
            IF QO_SELECT <> 'H'
                SET PRINT ON
                SET DEVICE TO PRINT

```

```

        IF QO_SELECT = 'J'
            @ 0, 1 SAY CHR(27) + CHR(15)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(77)
        ENDIF
        MAX_LINES = 66
    ELSE
        MAX_LINES = 23
    ENDIF
    IF (QO_SELECT <> 'J')
        SPACER = SPACE(17)
    ELSE
        SPACER = SPACE(48)
    ENDIF
    CLEAR
    @ 0, 0 SAY SPACER + 'DATE OF COMMISSIONING (DOC) FISCAL';
        + ' YEAR REPORT'
    @ 1, 0
    FIRST_TIME = .T.
    DISP_LINE = 2

*  vvvvvvvvvvv #8.  DATABASE RECORD LOOP  vvvvvvvvvvv *

DO WHILE (.NOT. EOF())
    IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
        IF (.NOT. FIRST_TIME)
            EJECT
        ENDIF
    ENDIF
    IF (FIRST_TIME)
        FIRST_TIME = .F.
    ELSE
        DISP_LINE = 0
        CLEAR
    ENDIF

*  vvvvvvvvvvvvvvv #9.  PAGING LOOP  vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))
    REC_NUM = RECNO()
    IF (DISP_LINE <= 3)
        @ DISP_LINE, 0 SAY HDR1A
        @ DISP_LINE + 1, 0 SAY HDR1B
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 1, 0 SAY SEP_LINE
        ENDIF
        DISP_LINE = DISP_LINE + 2
    ENDIF
    FTC = 'N'
    IF FT_COMP
        FTC = 'Y'
    ENDIF
    CLAS_VAL = AS_CLASS
    SELECT 2
    SEEK CLAS_VAL

```



```

        IF (.NOT. EOF())
            CLAS_NUM = STR(AS_CL_TOT,3)
        ELSE
            CLAS_NUM = ' ? '
        ENDIF
        SELECT 1
        GOTO REC_NUM
        @ DISP_LINE, 0 SAY &DATA1_S
        IF (QO_SELECT = 'J')
            @ DISP_LINE, 80 SAY &DATA1_L
        ENDIF
        DISP_LINE = DISP_LINE + 2

        * Issue dBASE III PLUS command to go to the      *
        * next record which meets the input constraints.*

        SKIP
    ENDDO

    * If the output media is the screen, issue the user*
    * paging prompt.                                     *

    IF (QO_SELECT = 'H')
        @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
        CLEAR TYPEAHEAD
        WAIT ''
    ENDIF
    ENDDO
    IF (QO_SELECT <> 'H')
        @ DISP_LINE + 1, 0 SAY CHR(10)
        EJECT
        IF (QO_SELECT = 'J')
            @ 0, 1 SAY CHR(18)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(80)
        ENDIF
        SET PRINT OFF
    ENDIF
    SET DEVICE TO SCREEN
    SET FILTER TO
ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure).                                *

ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 4 SAY 'PLEASE ENTER DATA.  PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ''
    @ 23, 0
    DONE = .F.

```

```

        ENDIF
    ENDIF
ENDDO
CLEAR

*   If the user has not previously entered a response to terminate the *
*   query (M_CHOICE would be "false"), then give them the opportunity *
*   to do another query or terminate the function.                      *

    IF (M_CHOICE)
        DO RCIS_HDR
        DO M_PROMPT
    ENDIF
ENDDO

*   Close the database files used in this query.  *

SELECT 2
USE
SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                                     *
*                                     *
*-----*
* SUMMARY:                           *
*     The CLAS_QRY procedure provides the interface for the user to per- *
*     form ad hoc queries on general cadet data which can be grouped by *
*     AS_CLASS, CAT_TYPE and PC_STATUS. *
*-----*

```

PROCEDURE CLAS\_QRY

\*

```

PRIVATE SPACER
PRIVATE MRM
PRIVATE MRE
PRIVATE MRF
PRIVATE WRK

```

\*

```

ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
CLEAR
M_CHOICE = .T.

```

```

* vvvvvvvvvvvvvvvvvvvvvvv #1.  MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvvvv *

```

DO WHILE (M\_CHOICE)

```

*   Initialize operator and constraint fields.   *

```

```

DONE = .F.
O1A = ' '
F1A = ' '
O1B = ' '
F1B = ' '
F2 = ' '
F3 = ' '
O4A = ' '
F4A = ' '
O4B = ' '
F4B = ' '
F5 = ' '

```

```

* vvvvvvvvvvvvvvvvvvvvvvv #2.  INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvvvv *

```

DO WHILE (.NOT. DONE)

CLEAR

DO HELP\_SCRN

@ 1, 0 TO 15,79

@ 1,17 SAY AIR SCIENCE CLASS GENERAL INFORMATION QUERY '

@ 3,28 SAY 'AS Class'

@ 6,23 SAY 'Category Type'

@ 8,16 SAY 'Pursuing/Conditional'

@ 10,27 SAY 'Last Name'

@ 13,32 SAY 'SSAN'

```

* vvvvvvvvvvvvvvvvv #3.  INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvv *

DO WHILE (.NOT. DONE)
  @ 3,37 GET O1A PICTURE '!!!'
  @ 3,40 GET F1A PICTURE '9'
  @ 4,37 GET O1B PICTURE '!!!'
  @ 4,40 GET F1B PICTURE '9'
  @ 6,40 GET F2 PICTURE '!'
  @ 8,40 GET F3 PICTURE '!'
  @ 10,37 GET O4A PICTURE '!!!'
  @ 10,40 GET F4A PICTURE '!!!!!!!!!!!!!!!!!!'
  @ 11,37 GET O4B PICTURE '!!!'
  @ 11,40 GET F4B PICTURE '!!!!!!!!!!!!!!!!!!'
  @ 13,40 GET F5 PICTURE '@R 999-99-9999'

  * Read query screen inputs and prepare to process them. *

  READ
  @ 23, 0
  @ 23,19 SAY ;
  "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
  CLEAR TYPEAHEAD
  READ

  * If the user chooses to cancel the query, set the required *
  * flags to terminate all procedure loops. *

  IF (DONE)
    STOP_LOOP = .T.
    M_CHOICE = .F.
    EXIT
  ELSE
    STOP_LOOP = .F.
  ENDIF
  @ 23, 0
  @ 23,19 SAY ;
  "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
  CLEAR TYPEAHEAD
  READ

  * If the user wants to change their inputs, set DONE flag to *
  * false and repeat the current loop. *

  IF (DONE)
    @ 23, 0
    DONE = .F.
    LOOP
  ELSE
    DONE = .T.
  ENDIF

* vvvvvvvvvvvvvvvvv #4.  RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvvvvv *

GOOD_RO = .T.

```

```

TEMP_LOOP = .T.
DO WHILE (TEMP_LOOP)
  IF (O1A <> ' ')
    DO RO_CHK WITH O1A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O1B <> ' ')
    DO RO_CHK WITH O1B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O4A <> ' ')
    DO RO_CHK WITH O4A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O4B <> ' ')
    DO RO_CHK WITH O4B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
  @ 23, 0
  ? CHR(7)
  M_CHOICE = .F.
  @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR.  WOULD YOU LIKE TO' ;
  ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'

  CLEAR TYPEAHEAD
  READ

  * Give the user the option of either returning to the      *
  * query input screen or terminating the query function.    *

  IF (M_CHOICE)
    @ 23, 0
    DONE = .F.
  ELSE
    STOP_LOOP = .T.
    EXIT
  ENDIF
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'.                                                  *

IF (STOP_LOOP)
  EXIT

```

ELSE

\* vvvvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvvvv \*

```
HDR1A = ''
HDR1B = ''
DATA1_S = ''
DATA1_L = ''
HDR1A = 'First           Last           AS      Cat           Purs';
      + '  Schl  Min    Min  Min'
HDR1B = 'Name           Name           Class Type Major Cond';
      + '  Type  Math  Eng  Fr1'
DATA1_S = "F_NAME+S2+L_NAME+S4+STR(AS_CLASS,1)+S5+CAT_TYPE+S4+MAJOR";
      + "+S4+PC_STATUS+S4+STR(SCHLR_TYPE,3,1)+S4+MRM+S5+MRE+S4+MRF+" ''
SEP_LINE = REPLICATE('-',80)
```

\*

```
IF (QO_SELECT = 'J')
  HDR1B = HDR1B + '  SSAN           Matric Work Corps Auxiliaries'
  DATA1_L = "S2+TRANSFORM(SSAN,'@R 999-99-9999')+S3+MATRIC+S3+WRK";
      + "+S4+TRANSFORM(CORPS_AUX,'@R !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!')";
  SEP_LINE = SEP_LINE + REPLICATE('-',57)
ENDIF
```

\* vvvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvvv \*

```
FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
  FILT_STR = 'AS_CLASS' + O1A + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
  ELSE
    FILT_STR = 'AS_CLASS' + O1B + F1B
  ENDIF
ENDIF
IF (LEN(LTRIM(F2)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.CAT_TYPE =' + "" + F2 + ""
  ELSE
    FILT_STR = 'CAT_TYPE =' + "" + F2 + ""
  ENDIF
ENDIF
IF (LEN(LTRIM(F3)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.PC_STATUS =' + "" + F3 + ""
  ELSE
    FILT_STR = 'PC_STATUS =' + "" + F3 + ""
  ENDIF
ENDIF
IF (LEN(LTRIM(F4A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O4A + "" + F4A + ""
  ELSE
    FILT_STR = 'L_NAME' + O4A + "" + F4A + ""
  ENDIF
ENDIF
```

```

ENDIF
ENDIF
IF (LEN(LTRIM(F4B)) > 0 .AND. (O4A <> O4B) .AND. (F4A <> F4B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O4B + "'" + F4B + "'"
  ELSE
    FILT_STR = 'L_NAME' + O4B + "'" + F4B + "'"
  ENDIF
ENDIF
ENDIF
IF (LEN(LTRIM(F5)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.SSAN =' + "'" + F5 + "'"
  ELSE
    FILT_STR = 'SSAN =' + "'" + F5 + "'"
  ENDIF
ENDIF
ENDIF
DONE = .T.

* vvvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvvv *

IF (LEN(FILT_STR) > 0)
  @ 23, 0
  @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
  SELECT 1
  IF (.NOT. FILE(M_NDX_F))
    INDEX ON &M_NDX_STR TO &M_NDX
  ENDIF
  SET INDEX TO &M_NDX
  SET FILTER TO &FILT_STR
  GOTO TOP
  DO CASE

    * If none of the database records meet all the input *
    * constraints, give the user the option to try again *
    * or to terminate the query. *

    CASE (EOF())
      DO ERR_NF
      IF (M_CHOICE)
        DONE = .F.
        LOOP
      ELSE
        EXIT
      ENDIF

    * If some database records meet the constraints, ini- *
    * tialize the print environment and perform print loop *
    * until all records are printed. *

    CASE (.NOT. EOF())
      IF QO_SELECT <> 'H'
        SET PRINT ON
        SET DEVICE TO PRINT
        IF QO_SELECT = 'J'
          @ 0, 1 SAY CHR(27) + CHR(15)

```

```

ELSE
    @ 0, 1 SAY CHR(27) + CHR(77)
ENDIF
MAX_LINES = 66
ELSE
    MAX_LINES = 23
ENDIF
IF (QO_SELECT <> 'J')
    SPACER = SPACE(18)
ELSE
    SPACER = SPACE(49)
ENDIF
CLEAR
@ 0, 0 SAY SPACER + 'AIR SCIENCE CLASS GENERAL';
    + ' INFORMATION REPORT'
@ 1, 0
FIRST_TIME = .T.
DISP_LINE = 2

* vvvvvvvvvvv #8. DATABASE RECORD LOOP vvvvvvvvvvv *

DO WHILE (.NOT. EOF())
    IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
        IF (.NOT. FIRST_TIME)
            EJECT
        ENDIF
    ENDIF
    IF (FIRST_TIME)
        FIRST_TIME = .F.
    ELSE
        DISP_LINE = 0
        CLEAR
    ENDIF

* vvvvvvvvvvvvvvv #9. PAGING LOOP vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))
    IF (DISP_LINE <= 3)
        @ DISP_LINE, 0 SAY HDR1A
        @ DISP_LINE + 1, 0 SAY HDR1B
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 1, 0 SAY SEP_LINE
        ENDIF
        DISP_LINE = DISP_LINE + 2
    ENDIF
    MRM = 'N'
    MRE = 'N'
    MRF = 'N'
    IF M_R_MATH
        MRM = 'Y'
    ENDIF
    IF M_R_ENGL
        MRE = 'Y'
    ENDIF
    IF M_R_FLAN

```



```

        MRF = 'Y'
    ENDIF
    @ DISP_LINE, 0 SAY &DATA1_S
    IF (QO_SELECT = 'J')
        WRK = 'N'
        IF WORK
            WRK = 'Y'
        ENDIF
        @ DISP_LINE, 80 SAY &DATA1_L
    ENDIF
    DISP_LINE = DISP_LINE + 2

    * Issue dBASE III PLUS command to go to the      *
    * next record which meets the input constraints.*

    SKIP
    ENDDO

    * If the output media is the screen, issue the user*
    * paging prompt.                                     *

    IF (QO_SELECT = 'H')
        @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
        CLEAR TYPEAHEAD
        WAIT ''
    ENDIF
    ENDDO
    IF (QO_SELECT <> 'H')
        @ DISP_LINE + 1, 0 SAY CHR(10)
        EJECT
        IF (QO_SELECT = 'J')
            @ 0, 1 SAY CHR(18)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(80)
        ENDIF
        SET PRINT OFF
    ENDIF
    SET DEVICE TO SCREEN
    SET FILTER TO
ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure).                                *

ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 4 SAY 'PLEASE ENTER DATA.  PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ''
    @ 23, 0
    DONE = .F.
ENDIF

```

```

        ENDIF
    ENDDO
    CLEAR

    * If the user has not previously entered a response to terminate the *
    * query (M_CHOICE would be "false"), then give them the opportunity *
    * to do another query or terminate the function. *

    IF (M_CHOICE)
        DO RCIS_HDR
        DO M_PROMPT
    ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                HRAX_QRY                *
*-----*
*
* SUMMARY:
*      The HRAX_QRY procedure provides the interface for the user to per-
*      form ad hoc queries on required cadet data for two-year program
*      candidates and additional data related to the horizontal axis.
*
*-----*

```

```
PROCEDURE HRAX_QRY
```

```
*
```

```
PRIVATE ALT
```

```
PRIVATE SPACER
```

```
*
```

```
ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
```

```
CLEAR
```

```
M_CHOICE = .T.
```

```
* vvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvv *
```

```
DO WHILE (M_CHOICE)
```

```
    * Initialize operator and constraint fields. *
```

```
    DONE = .F.
```

```
    O1A = ' '
```

```
    F1A = ' '
```

```
    O1B = ' '
```

```
    F1B = ' '
```

```
    F2 = ' '
```

```
    O3A = ' '
```

```
    F3A = ' '
```

```
    O3B = ' '
```

```
    F3B = ' '
```

```
    F4 = ' '
```

```
* vvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvv *
```

```
DO WHILE (.NOT. DONE)
```

```
    CLEAR
```

```
    DO HELP_SCRN
```

```
    @ 3, 0 TO 15,79
```

```
    @ 3,14 SAY ' TWO-YEAR PROGRAM CANDIDATE (HORIZONTAL AXIS) QUERY '
```

```
    @ 5,28 SAY 'AS Class'
```

```
    @ 8,23 SAY 'Category Type'
```

```
    @ 10,27 SAY 'Last Name'
```

```
    @ 13,32 SAY 'SSAN'
```

```
* vvvvvvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvvvv *
```

```
DO WHILE (.NOT. DONE)
```

```
    @ 5,37 GET O1A PICTURE '!!!'
```

```

@ 5,40 GET F1A PICTURE '9'
@ 6,37 GET O1B PICTURE '!!'
@ 6,40 GET F1B PICTURE '9'
@ 8,40 GET F2 PICTURE '!'
@ 10,37 GET O3A PICTURE '!!'
@ 10,40 GET F3A PICTURE '!!!!!!!!!!!!!!'
@ 11,37 GET O3B PICTURE '!!'
@ 11,40 GET F3B PICTURE '!!!!!!!!!!!!!!'
@ 13,40 GET F4 PICTURE '@R 999-99-9999'

```

\* Read query screen inputs and prepare to process them. \*

```

READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

\* If the user chooses to cancel the query, set the required \*  
 \* flags to terminate all procedure loops. \*

```

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE = .F.
  EXIT
ELSE
  STOP_LOOP = .F.
ENDIF
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

\* If the user wants to change their inputs, set DONE flag to \*  
 \* false and repeat the current loop. \*

```

IF (DONE)
  @ 23, 0
  DONE = .F.
  LOOP
ELSE
  DONE = .T.
ENDIF

```

\* vvvvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvv \*

```

GOOD_RO = .T.
TEMP_LOOP = .T.
DO WHILE (TEMP_LOOP)
  IF (O1A <> ' ')
    DO RO_CHK WITH O1A
    IF (.NOT. GOOD_RO)
      EXIT

```

```

        ENDIF
    ENDIF
    IF (O1B <> ' ')
        DO RO_CHK WITH O1B
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (O3A <> ' ')
        DO RO_CHK WITH O3A
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    IF (O3B <> ' ')
        DO RO_CHK WITH O3B
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
    @ 23, 0
    ? CHR(7)
    M_CHOICE = .F.
    @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR. WOULD YOU LIKE TO' ;
        + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
    CLEAR TYPEAHEAD
    READ

    * Give the user the option of either returning to the      *
    * query input screen or terminating the query function.    *

    IF (M_CHOICE)
        @ 23, 0
        DONE = .F.
    ELSE
        STOP_LOOP = .T.
        EXIT
    ENDIF
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'.                                                *

IF (STOP_LOOP)
    EXIT
ELSE

* vvvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvvv *

HDR1A = ''
HDR1B = ''

```

```

HDR2A = ''
HDR2B = ''
DATA1_S = ''
DATA1_L = ''
DATA2_S = ''
DATA2_L = ''
SEP_LINE = ''
BLK_LINE = ''
HDR1A = 'First          Last          AS    Cat    Phys  ';
      + 'Physical
HDR1B = 'Name          Name          Class Type Cat  ';
      + 'Date          ALTU Race
DATA1_S = "F_NAME+S2+L_NAME+S4+STR(AS_CLASS,1)+S5+CAT_TYPE+S5";
      + "+PHY_CAT+S4+DLOC(PHY_DATE)+S3+ALT+S5+RACE+S7+' '"
HDR2A = '          AFOQT          SAT  ';
      + '          GPA          DC
HDR2B = '          Quan Verb Pil Nav AcAp Cum Math';
      + ' Verb Cum Sem Rtngr

*
DATA2A = "S17+' '+STR(AFOQT_QUAN,2)+S5+STR(AFOQT_VERB,2)+S3";
      + "+STR(AFOQT_PLT,2)+S3+STR(AFOQT_NAV,2)+S4+STR(AFOQT_AA,2)";
      + "+S3+STR(SAT_CUM,4)+S3+STR(SAT_MATH,3)+S3+STR(SAT_VERB,3)";
DATA2B = "+S2+STR(CUM_GPA,4,2)+S2+STR(SEM_GPA,4,2)+S3";
      + "+STR(DC_RTNG,1)+S2"
DATA2_S = DATA2A + DATA2B

*
SEP_LINE = REPLICATE('-',80)
BLK_LINE = REPLICATE('-',80)
SQG_LINE = REPLICATE('~',80)

*
IF (QO_SELECT = 'J')
  HDR1A = HDR1A + ' LOCAL'
  HDR1B = HDR1B + ' Street          City          Zip  ';
      + ' Phone'
  DATA1_L = "S2+LOCAL_STRT+S2+LEFT(LOCAL_CITY,15)+S2";
      + "+LEFT(LOCAL_ZIP,5)+S2+TRANSFORM(LOCAL_PHON,'@R 999-9999')";
  HDR2A = HDR2A + ' ACT          Form 48'
  HDR2B = HDR2B + ' Cum Math Engl NSci SSci Date'
  DATA2_L = "S3+STR(ACT_CUM,2)+S3+STR(ACT_MATH,2)+S4";
      + "+STR(ACT_ENGL,2)+S4+STR(ACT_NSCL,2)+S4";
      + "+STR(ACT_SSCI,2)+S4+DLOC(FORM_48)"
  SEP_LINE = SEP_LINE + REPLICATE('-',57)
  BLK_LINE = BLK_LINE + REPLICATE('-',57)
  SQG_LINE = SQG_LINE + REPLICATE('~',57)
ENDIF

* vvvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvvv *

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
  FILT_STR = 'AS_CLASS' + O1A + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
  
```

```

ELSE
    FILT_STR = 'AS_CLASS' + 01B + F1B
ENDIF
ENDIF
IF (LEN(LTRIM(F2)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.CAT_TYPE =' + ''' + F2 + '''
    ELSE
        FILT_STR = 'CAT_TYPE =' + ''' + F2 + '''
    ENDIF
ENDIF
IF (LEN(LTRIM(F3A)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.L_NAME' + 03A + ''' + F3A + '''
    ELSE
        FILT_STR = 'L_NAME' + 03A + ''' + F3A + '''
    ENDIF
ENDIF
IF (LEN(LTRIM(F3B)) > 0 .AND. (03A <> 03B) .AND. (F3A <> F3B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.L_NAME' + 03B + ''' + F3B + '''
    ELSE
        FILT_STR = 'L_NAME' + 03B + ''' + F3B + '''
    ENDIF
ENDIF
IF (LEN(LTRIM(F4)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.SSAN =' + ''' + F4 + '''
    ELSE
        FILT_STR = 'SSAN =' + ''' + F4 + '''
    ENDIF
ENDIF
DONE = .T.

* vvvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvv *

IF (LEN(FILT_STR) > 0)
    @ 23, 0
    @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
    SELECT 1
    IF (.NOT. FILE(M_NDX_F))
        INDEX ON &M_NDX_STR TO &M_NDX
    ENDIF
    SET INDEX TO &M_NDX
    SET FILTER TO &FILT_STR
    GOTO TOP
    DO CASE

        * If none of the database records meet all the input *
        * constraints, give the user the option to try again *
        * or to terminate the query. *

        CASE (EOF())
            DO ERR_NF
            IF (M_CHOICE)

```

```

        DONE = .F.
        LOOP
    ELSE
        EXIT
    ENDIF

*   If some database records meet the constraints, ini- *
*   tialize the print environment and perform print loop *
*   until all records are printed.                        *

CASE (.NOT. EOF())
    IF QO_SELECT <> 'H'
        SET PRINT ON
        SET DEVICE TO PRINT
        IF QO_SELECT = 'J'
            @ 0, 1 SAY CHR(27) + CHR(15)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(77)
        ENDIF
        MAX_LINES = 66
    ELSE
        MAX_LINES = 23
    ENDIF
    IF (QO_SELECT <> 'J')
        SPACER = SPACE(15)
    ELSE
        SPACER = SPACE(46)
    ENDIF
    CLEAR
    @ 0, 0 SAY SPACER + 'TWO-YEAR PROGRAM CANDIDATE';
        + ' (HORIZONTAL AXIS) REPORT'
    @ 1, 0
    FIRST_TIME = .T.
    DISP_LINE = 2

*   vvvvvvvvvvv #8.  DATABASE RECORD LOOP  vvvvvvvvvvv *

DO WHILE (.NOT. EOF())
    IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
        IF (.NOT. FIRST_TIME)
            EJECT
        ENDIF
    ENDIF
    IF (FIRST_TIME)
        FIRST_TIME = .F.
    ELSE
        DISP_LINE = 0
        CLEAR
    ENDIF

*   vvvvvvvvvvvvvvv #9.  PAGING LOOP  vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))

    *   If the number of print lines per cadet will   *

```



```
* not fit on one page, exit the loop and go to *
* the next page. *
```

```
IF ((MAX_LINES - DISP_LINE) < 7)
EXIT
ELSE
@ DISP_LINE, 0 SAY HDR1A
@ DISP_LINE + 1, 0 SAY HDR1B
IF (QO_SELECT <> 'H')
@ DISP_LINE + 1, 0 SAY SEP_LINE
ENDIF
ALT = 'N'
IF ALTU
ALT = 'Y'
ENDIF
@ DISP_LINE + 2, 0 SAY &DATA1_S
IF (QO_SELECT = 'J')
@ DISP_LINE + 2, 80 SAY &DATA1_L
ENDIF
@ DISP_LINE + 4, 0 SAY HDR2A
@ DISP_LINE + 5, 0 SAY HDR2B
IF (QO_SELECT <> 'H')
SEP_LINE = STUFF(SEP_LINE,1,17,S17)
@ DISP_LINE + 5, 0 SAY SEP_LINE
SEP_LINE = STUFF(SEP_LINE,1,17,REPLICATE('_',17))
ENDIF
DL = DISP_LINE + 6
```

```
* The position of the following line is critical for it to print properly. *
* The string variable is so long that DOS will not accept it unless it is *
* <= 256 characters when combined with the other commands on the same line.*
```

```
*****
@ DL, 0 SAY &DATA2_S
*****
```

```
IF (QO_SELECT = 'J')
@ DISP_LINE + 6, 80 SAY &DATA2_L
ENDIF
@ DISP_LINE + 7, 0 SAY SQG_LINE
DISP_LINE = DISP_LINE + 8
ENDIF
```

```
* Issue dBASE III PLUS command to go to the *
* next record which meets the input constraints.*
```

```
SKIP
ENDDO
```

```
* If the output media is the screen, issue the user*
* paging prompt. *
```

```
IF (QO_SELECT = 'H')
@ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
CLEAR TYPEAHEAD
WAIT ''
```

```

        ENDIF
    ENDDO
    IF (QO_SELECT <> 'H')
        @ DISP_LINE + 1, 0 SAY CHR(10)
        EJECT
        IF (QO_SELECT = 'J')
            @ 0, 1 SAY CHR(18)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(80)
        ENDIF
        SET PRINT OFF
    ENDIF
    SET DEVICE TO SCREEN
    SET FILTER TO

ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure). *

ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 4 SAY 'PLEASE ENTER DATA.  PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ''
    @ 23, 0
    DONE = .F.
ENDIF

ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *
* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
    DO RCIS_HDR
    DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                                CGDT_QRY                                *
*-----*
*
* SUMMARY:
*      The CGDT_QRY procedure provides the interface for the user to per-
*      form ad hoc queries on cadet data which is related to suspense
*      dates pertaining to their graduation and their commissioning.
*
*-----*

```

# ``` PROCEDURE CGDT_QRY ```

```
*
```

```
PRIVATE SPACER
```

```
*
```

```
ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
```

```
CLEAR
```

```
M_CHOICE = .T.
```

```
* vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvv *
```

```
DO WHILE (M_CHOICE)
```

```
    * Initialize operator and constraint fields. *
```

```
DONE = .F.
```

```
O1A = ' '
```

```
F1A = ' '
```

```
O1B = ' '
```

```
F1B = ' '
```

```
O2A = ' '
```

```
F2A = ' '
```

```
O2B = ' '
```

```
F2B = ' '
```

```
F3 = ' '
```

```
O4A = ' '
```

```
F4A = ' '
```

```
O4B = ' '
```

```
F4B = ' '
```

```
O5A = ' '
```

```
F5A = ' '
```

```
O5B = ' '
```

```
F5B = ' '
```

```
* vvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvv *
```

```
DO WHILE (.NOT. DONE)
```

```
    CLEAR
```

```
    DO HELP_SCRN
```

```
    @ 5, 0 TO 15,79
```

```
    @ 5,17 SAY ' GRADUATION/COMMISSIONING SUSPENSE DATES QUERY '
```

```
    @ 7, 9 SAY 'AS Class'
```

```
    @ 10,11 SAY 'Last Name'
```

```
    @ 13,16 SAY 'SSAN'
```

```

@ 7,47 SAY '# Days Until'
@ 8,47 SAY 'Commissioning Date'
@ 10,50 SAY '# Days Until'
@ 11,50 SAY 'Graduation Date'

```

```

* vvvvvvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvvvvvv *

```

```

DO WHILE (.NOT. DONE)
@ 7,21 GET O1A PICTURE '!!'
@ 7,24 GET F1A PICTURE '9'
@ 8,21 GET O1B PICTURE '!!'
@ 8,24 GET F1B PICTURE '9'
@ 10,21 GET O2A PICTURE '!!'
@ 10,24 GET F2A PICTURE '!!!!!!!!!!!!!!!!!!!!'
@ 11,21 GET O2B PICTURE '!!'
@ 11,24 GET F2B PICTURE '!!!!!!!!!!!!!!!!!!!!'
@ 13,24 GET F3 PICTURE '@R 999-99-9999'
@ 7,66 GET O4A PICTURE '!!'
@ 7,69 GET F4A PICTURE '999'
@ 8,66 GET O4B PICTURE '!!'
@ 8,69 GET F4B PICTURE '999'
@ 10,66 GET O5A PICTURE '!!'
@ 10,69 GET F5A PICTURE '999'
@ 11,66 GET O5B PICTURE '!!'
@ 11,69 GET F5B PICTURE '999'

```

```

* Read query screen inputs and prepare to process them. *

```

```

READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

```

* If the user chooses to cancel the query, set the required *
* flags to terminate all procedure loops. *

```

```

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE = .F.
  EXIT

```

```

ELSE
  STOP_LOOP = .F.
ENDIF

```

```

@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

```

```

* If the user wants to change their inputs, set DONE flag to *
* false and repeat the current loop. *

```

```

IF (DONE)

```

```

@ 23, 0
DONE = .F.
LOOP
ELSE
  DONE = .T.
ENDIF

```

```

* vvvvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvv *

```

```

GOOD_RO = .T.
TEMP_LOOP = .T.
DO WHILE (TEMP_LOOP)
  IF (O1A <> ' ')
    DO RO_CHK WITH O1A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O1B <> ' ')
    DO RO_CHK WITH O1B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O2A <> ' ')
    DO RO_CHK WITH O2A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O2B <> ' ')
    DO RO_CHK WITH O2B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O4A <> ' ')
    DO RO_CHK WITH O4A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O4B <> ' ')
    DO RO_CHK WITH O4B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O5A <> ' ')
    DO RO_CHK WITH O5A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O5B <> ' ')

```

```

        DO RO_CHK WITH 05B
        IF (.NOT. GOOD_RO)
            EXIT
        ENDIF
    ENDIF
    TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
    @ 23, 0
    ? CHR(7)
    M_CHOICE = .F.
    @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR. WOULD YOU LIKE TO'
    @ 23,52 SAY ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
    CLEAR TYPEAHEAD
    READ

    * Give the user the option of either returning to the      *
    * query input screen or terminating the query function.    *

    IF (M_CHOICE)
        @ 23, 0
        DONE = .F.
    ELSE
        STOP_LOOP = .T.
        EXIT
    ENDIF
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'. *

IF (STOP_LOOP)
    EXIT
ELSE

* vvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvv *

HDR1A = ''
HDR1B = ''
DATA1_S = ''
DATA1_L = ''
HDR1A = 'First          Last          Comm          Grad          ';
      + ' AS
HDR1B = 'Name          Name          Date          Date          ';
      + 'Class    SSAN
DATA1_S = "F_NAME+S2+L_NAME+S3+DTC( COM_DATE)+S3+DTC( GRAD_DATE)+S4";
      + "+STR(AS_CLASS,1)+S6+TRANSFORM(SSAN,'@R 999-99-9999')+S4"
SEP_LINE = REPLICATE('-',80)

IF (QO_SELECT = 'J')
    DATA1_L = "S2"
    SEP_LINE = SEP_LINE + REPLICATE('-',52)
ENDIF

```

\* vvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvv \*

```
FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
  FILT_STR = 'AS_CLASS' + O1A + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
  ELSE
    FILT_STR = 'AS_CLASS' + O1B + F1B
  ENDIF
ENDIF
IF (LEN(LTRIM(F2A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O2A + "'" + F2A + "'"
  ELSE
    FILT_STR = 'L_NAME' + O2A + "'" + F2A + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F2B)) > 0 .AND. (O2A <> O2B) .AND. (F2A <> F2B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O2B + "'" + F2B + "'"
  ELSE
    FILT_STR = 'L_NAME' + O2B + "'" + F2B + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F3)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.SSAN =' + "'" + F3 + "'"
  ELSE
    FILT_STR = 'SSAN =' + "'" + F3 + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F4A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.(COM_DATE-DATE())' + O4A + F4A
  ELSE
    FILT_STR = '(COM_DATE-DATE())' + O4A + F4A
  ENDIF
ENDIF
IF (LEN(LTRIM(F4B)) > 0 .AND. (O4A <> O4B) .AND. (F4A <> F4B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.(COM_DATE-DATE())' + O4B + F4B
  ELSE
    FILT_STR = '(COM_DATE-DATE())' + O4B + F4B
  ENDIF
ENDIF
IF (LEN(LTRIM(F5A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.(GRAD_DATE-DATE())' + O5A + F5A
  ELSE
    FILT_STR = '(GRAD_DATE-DATE())' + O5A + F5A
  ENDIF
ENDIF
```

```

IF (LEN(LTRIM(F5B)) > 0 .AND. (O5A <> O5B) .AND. (F5A <> F5B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.(GRAD_DATE-DATE())' + O5B + F5B
  ELSE
    FILT_STR = '(GRAD_DATE-DATE())' + O5B + F5B
  ENDIF
ENDIF
DONE = .T.

* vvvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvvv *

IF (LEN(FILT_STR) > 0)
  @ 23, 0
  @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
  SELECT 1
  IF (.NOT. FILE(M_NDX_F))
    INDEX ON &M_NDX_STR TO &M_NDX
  ENDIF
  SET INDEX TO &M_NDX
  SET FILTER TO &FILT_STR
  GOTO TOP
DO CASE

  * If none of the database records meet all the input *
  * constraints, give the user the option to try again *
  * or to terminate the query. *

  CASE (EOF())
    DO ERR_NF
    IF (M_CHOICE)
      DONE = .F.
      LOOP
    ELSE
      EXIT
    ENDIF

  * If some database records meet the constraints, ini- *
  * tialize the print environment and perform print loop *
  * until all records are printed. *

  CASE (.NOT. EOF())
    IF QO_SELECT <> 'H'
      SET PRINT ON
      SET DEVICE TO PRINT
      IF QO_SELECT = 'J'
        @ 0, 1 SAY CHR(27) + CHR(15)
      ELSE
        @ 0, 1 SAY CHR(27) + CHR(77)
      ENDIF
      MAX_LINES = 66
    ELSE
      MAX_LINES = 23
    ENDIF
    IF (QO_SELECT <> 'J')
      SPACER = SPACE(17)

```



```

ELSE
    SPACER = SPACE(48)
ENDIF
CLEAR
@ 0, 0 SAY SPACER + 'GRADUATION/COMMISSIONING SUSPENSE';
    + ' DATES REPORT'
@ 1, 0
FIRST_TIME = .T.
DISP_LINE = 2

* vvvvvvvvvvv #8. DATABASE RECORD LOOP vvvvvvvvvvv *

DO WHILE (.NOT. EOF())
    IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
        IF (.NOT. FIRST_TIME)
            EJECT
        ENDIF
    ENDIF
    IF (FIRST_TIME)
        FIRST_TIME = .F.
    ELSE
        DISP_LINE = 0
        CLEAR
    ENDIF

* vvvvvvvvvvvvvvv #9. PAGING LOOP vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))
    IF (DISP_LINE <= 3)
        @ DISP_LINE, 0 SAY HDR1A
        @ DISP_LINE + 1, 0 SAY HDR1B
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 1, 0 SAY SEP_LINE
        ENDIF
        DISP_LINE = DISP_LINE + 2
    ENDIF
    @ DISP_LINE, 0 SAY &DATA1_S
    IF (QO_SELECT = 'J')
        @ DISP_LINE, 80 SAY &DATA1_L
    ENDIF
    DISP_LINE = DISP_LINE + 2

    * Issue dBASE III PLUS command to go to the *
    * next record which meets the input constraints.*

    SKIP
ENDDO

* If the output media is the screen, issue the user*
* paging prompt. *

IF (QO_SELECT = 'H')
    @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
    CLEAR TYPEAHEAD
    WAIT ''

```

```

        ENDIF
    ENDDO
    IF (QO_SELECT <> 'H')
        @ DISP_LINE + 1, 0 SAY CHR(10)
        EJECT
        IF (QO_SELECT = 'J')
            @ -0, 1 SAY CHR(18)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(80)
        ENDIF
        SET PRINT OFF
    ENDIF
    SET DEVICE TO SCREEN
    SET FILTER TO

ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure). *

ELSE
    @ 23, 0
    ? CHR(7)
    @ 23, 4 SAY 'PLEASE ENTER DATA.  PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ''
    @ 23, 0
    DONE = .F.
ENDIF
ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *
* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
    DO RCIS_HDR
    DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                               SEDT_QRY                               *
*-----*
*
* SUMMARY:
*   The SEDT_QRY procedure provides the interface for the user to per-
*   form ad hoc queries on cadet data which is related to the cadet's
*   scholarship expiration date (if they have one), i.e. suspense dates
*
*-----*

```

# ``` PROCEDURE SEDT_QRY ```

```

*
* PRIVATE SPACER
*
ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
CLEAR
M_CHOICE = .T.

* vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvvvv *

DO WHILE (M_CHOICE)

* Initialize operator and constraint fields. *

DONE = .F.
O1A = ' '
F1A = ' '
O1B = ' '
F1B = ' '
F2 = ' '
O3A = ' '
F3A = ' '
O3B = ' '
F3B = ' '
O4A = ' '
F4A = ' '
O4B = ' '
F4B = ' '
F5 = ' '

* vvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvvvv *

DO WHILE (.NOT. DONE)
  CLEAR
  DO HELP_SCRN
  @ 1, 0 TO 15,79
  @ 1,22 SAY ' SCHOLARSHIP EXPIRATION DATES QUERY '
  @ 3,28 SAY 'AS Class'
  @ 6,23 SAY 'Category Type'
  @ 8,16 SAY 'Scholarship Type'
  @ 11,27 SAY 'Last Name'
  @ 14,32 SAY 'SSAN'

```

\* vvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvv \*

DO WHILE (.NOT. DONE)

```
@ 3,37 GET O1A PICTURE '!!'
@ 3,40 GET F1A PICTURE '9'
@ 4,37 GET O1B PICTURE '!!'
@ 4,40 GET F1B PICTURE '9'
@ 6,40 GET F2 PICTURE '!'
@ 8,37 GET O3A PICTURE '!!'
@ 8,40 GET F3A PICTURE '9.9'
@ 9,37 GET O3B PICTURE '!!'
@ 9,40 GET F3B PICTURE '9.9'
@ 11,37 GET O4A PICTURE '!!'
@ 11,40 GET F4A PICTURE '!!!!!!!!!!!!!!!!'
@ 12,37 GET O4B PICTURE '!!'
@ 12,40 GET F4B PICTURE '!!!!!!!!!!!!!!!!'
@ 14,40 GET F5 PICTURE '@R 999-99-9999'
```

\* Read query screen inputs and prepare to process them. \*

READ

@ 23, 0

@ 23,19 SAY ;

"DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'

CLEAR TYPEAHEAD

READ

\* If the user chooses to cancel the query, set the required \*  
\* flags to terminate all procedure loops. \*

IF (DONE)

STOP\_LOOP = .T.

M\_CHOICE = .F.

EXIT

ELSE

STOP\_LOOP = .F.

ENDIF

@ 23, 0

@ 23,19 SAY ;

"DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'

CLEAR TYPEAHEAD

READ

\* If the user wants to change their inputs, set DONE flag to \*  
\* false and repeat the current loop. \*

IF (DONE)

@ 23, 0

DONE = .F.

LOOP

ELSE

DONE = .T.

ENDIF

\* vvvvvvvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvvvvv \*

```

GOOD_RO    = .T.
TEMP_LOOP  = .T.
DO WHILE (TEMP_LOOP)
  IF (O1A <> ' ')
    DO RO_CHK WITH O1A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O1B <> ' ')
    DO RO_CHK WITH O1B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O3A <> ' ')
    DO RO_CHK WITH O3A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O3B <> ' ')
    DO RO_CHK WITH O3B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O4A <> ' ')
    DO RO_CHK WITH O4A
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  IF (O4B <> ' ')
    DO RO_CHK WITH O4B
    IF (.NOT. GOOD_RO)
      EXIT
    ENDIF
  ENDIF
  TEMP_LOOP = .F.
ENDDO
IF (.NOT. GOOD_RO)
  @ 23, 0
  ? CHR(7)
  M_CHOICE = .F.
  @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR.  WOULD YOU LIKE TO' ;
    + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
  CLEAR TYPEAHEAD
  READ

  * Give the user the option of either returning to the      *
  * query input screen or terminating the query function.    *

  IF (M_CHOICE)

```

```

        @ 23, 0
        DONE = .F.
    ELSE
        STOP_LOOP = .T.
        EXIT
    ENDIF
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'. *

IF (STOP_LOOP)
    EXIT
ELSE

* vvvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvvv *

HDR1A = ''
HDR1B = ''
DATA1_S = ''
DATA1_L = ''
HDR1A = 'First           Last           Schl Exp  Sch  Corps      ';
      + '           Semester'
HDR1B = 'Name           Name           Date      Typ  Position  ';
      + '           Intrview'
DATA1_S = "LEFT(F_NAME,14)+S2+LEFT(L_NAME,14)+S2+DLOC(SCHLR_DATE)";
      + "+S2+STR(SCHLR_TYPE,3,1)+S2+LEFT(CORPS_POS,23)+S2";
      + "+DLOC(SEM_INTRVW)"
SEP_LINE = REPLICATE('_',80)

*

IF (QO_SELECT = 'J')
    HDR1A = HDR1A + ' Significant'
    HDR1B = HDR1B + ' Information'
    DATA1_L = "S2+OTHER_INFO"
    SEP_LINE = SEP_LINE + REPLICATE('_',52)
ENDIF

* vvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvv *

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
    FILT_STR = 'AS_CLASS' + O1A + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
    ELSE
        FILT_STR = 'AS_CLASS' + O1B + F1B
    ENDIF
ENDIF
IF (LEN(LTRIM(F2)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.CAT_TYPE =' + "" + F2 + ""
    ELSE

```

```

        FILT_STR = 'CAT_TYPE =' + ' ' + F2 + ' '
    ENDIF
ENDIF
IF (LEN(LTRIM(F3A)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.SCHLR_TYPE' + O3A + F3A
    ELSE
        FILT_STR = 'SCHLR_TYPE' + O3A + F3A
    ENDIF
ENDIF
IF (LEN(LTRIM(F3B)) > 0 .AND. (O3A <> O3B) .AND. (F3A <> F3B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.SCHLR_TYPE' + O3B + F3B
    ELSE
        FILT_STR = 'SCHLR_TYPE' + O3B + F3B
    ENDIF
ENDIF
IF (LEN(LTRIM(F4A)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.L_NAME' + O4A + ' ' + F4A + ' '
    ELSE
        FILT_STR = 'L_NAME' + O4A + ' ' + F4A + ' '
    ENDIF
ENDIF
IF (LEN(LTRIM(F4B)) > 0 .AND. (O4A <> O4B) .AND. (F4A <> F4B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.L_NAME' + O4B + ' ' + F4B + ' '
    ELSE
        FILT_STR = 'L_NAME' + O4B + ' ' + F4B + ' '
    ENDIF
ENDIF
IF (LEN(LTRIM(F5)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.SSAN =' + ' ' + F5 + ' '
    ELSE
        FILT_STR = 'SSAN =' + ' ' + F5 + ' '
    ENDIF
ENDIF
DONE = .T.

```

\* vvvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvvv \*

```

IF (LEN(FILT_STR) > 0)
    @ 23, 0
    @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
    SELECT 1
    IF (.NOT. FILE(M_NDX_F))
        INDEX ON &M_NDX_STR TO &M_NDX
    ENDIF
    SET INDEX TO &M_NDX
    SET FILTER TO &FILT_STR
    GOTO TOP
DO CASE

```

\* If none of the database records meet all the input \*

```

* constraints, give the user the option to try again *
* or to terminate the query. *

```

```

CASE (EOF())
  DO ERR_NF
  IF (M_CHOICE)
    DONE = .F.
    LOOP
  ELSE
    EXIT
  ENDIF

```

```

* If some database records meet the constraints, ini- *
* tialize the print environment and perform print loop *
* until all records are printed. *

```

```

CASE (.NOT. EOF())
  IF QQ_SELECT <> 'H'
    SET PRINT ON
    SET DEVICE TO PRINT
    IF QQ_SELECT = 'J'
      @ 0, 1 SAY CHR(27) + CHR(15)
    ELSE
      @ 0, 1 SAY CHR(27) + CHR(77)
    ENDIF
    MAX_LINES = 66
  ELSE
    MAX_LINES = 23
  ENDIF
  IF (QQ_SELECT <> 'J')
    SPACER = SPACE(22)
  ELSE
    SPACER = SPACE(53)
  ENDIF
  CLEAR
  @ 0, 0 SAY SPACER + 'SCHOLARSHIP EXPIRATION DATES REPORT'
  @ 1, 0
  FIRST_TIME = .T.
  DISP_LINE = 2

```

```

* vvvvvvvvvvv #8. DATABASE RECORD LOOP vvvvvvvvvvv *

```

```

DO WHILE (.NOT. EOF())
  IF ((DISP_LINE > 0) .AND. (QQ_SELECT <> 'H'))
    IF (.NOT. FIRST_TIME)
      EJECT
    ENDIF
  ENDIF
  IF (FIRST_TIME)
    FIRST_TIME = .F.
  ELSE
    DISP_LINE = 0
    CLEAR
  ENDIF

```



```

* vvvvvvvvvvvvvvv #9. PAGING LOOP vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))
  IF (DISP_LINE <= 3)
    @ DISP_LINE, 0 SAY HDR1A
    @ DISP_LINE + 1, 0 SAY HDR1B
    IF (QO_SELECT <> 'H')
      @ DISP_LINE + 1, 0 SAY SEP_LINE
    ENDIF
    DISP_LINE = DISP_LINE + 2
  ENDIF
  @ DISP_LINE, 0 SAY &DATA1_S
  IF (QO_SELECT = 'J')
    @ DISP_LINE, 80 SAY &DATA1_L
  ENDIF
  DISP_LINE = DISP_LINE + 2

  * Issue dBASE III PLUS command to go to the      *
  * next record which meets the input constraints.*

  SKIP
ENDDO

* If the output media is the screen, issue the user*
* paging prompt.                                     *

IF (QO_SELECT = 'H')
  @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
  CLEAR TYPEAHEAD
  WAIT ''
ENDIF
ENDDO
IF (QO_SELECT <> 'H')
  @ DISP_LINE + 1, 0 SAY CHR(10)
  EJECT
  IF (QO_SELECT = 'J')
    @ 0, 1 SAY CHR(18)
  ELSE
    @ 0, 1 SAY CHR(27) + CHR(80)
  ENDIF
  SET PRINT OFF
ENDIF
SET DEVICE TO SCREEN
SET FILTER TO

ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure).                                *

ELSE
  @ 23, 0
  ? CHR(7)
  @ 23, 4 SAY 'PLEASE ENTER DATA. PRESS ANY KEY TO CONTINUE.'

```

```

        CLEAR TYPEAHEAD
        WAIT ''
        @ 23, 0
        DONE = .F.
    ENDF
ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *
* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
    DO RCIS_HDR
    DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                               WTAR_QRY                               *
*-----*
* SUMMARY:                                                              *
*   The WTAR_QRY procedure provides the interface for the user to per- *
*   form ad hoc queries on cadet data which is related to the cadet's  *
*   weight and aerobic run time standards.                             *
*-----*

```

```

PROCEDURE WTAR_QRY

```

```

*
PRIVATE PRINT_OPT
PRIVATE PRNT_FLAG
PRIVATE SPACER

```

```

*
ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
CLEAR
M_CHOICE = .T.

```

```

* vvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvv *

```

```

DO WHILE (M_CHOICE)

```

```

    * Initialize operator and constraint fields. *

```

```

    DONE = .F.

```

```

    O1A = ' '

```

```

    F1A = ' '

```

```

    O1B = ' '

```

```

    F1B = ' '

```

```

    O2A = ' '

```

```

    F2A = ' '

```

```

    O2B = ' '

```

```

    F2B = ' '

```

```

    F3 = ' '

```

```

    PRINT_OPT = 1

```

```

* vvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvv *

```

```

DO WHILE (.NOT. DONE)

```

```

    CLEAR

```

```

    DO HELP_SCRN

```

```

    @ 1, 0 TO 15,79

```

```

    @ 1,19 SAY ' CADET WEIGHT AND AEROBIC STANDARDS QUERY '

```

```

    @ 3,23 SAY 'AS Class'

```

```

    @ 6,22 SAY 'Last Name'

```

```

    @ 9,27 SAY 'SSAN'

```

```

    @ 11,18 SAY 'Print Options'

```

```

    @ 12,18 SAY ' *Subject to constraints above*'

```

```

    @ 13,18 SAY ' All Cadets - 1'

```

```

    @ 14,18 SAY ' Only Cadets in violation of standards - 2'

```

\* vvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvv \*

```
DO WHILE (.NOT. DONE)
  @ 3,32 GET O1A PICTURE '!!'
  @ 3,35 GET F1A PICTURE '9'
  @ 4,32 GET O1B PICTURE '!!'
  @ 4,35 GET F1B PICTURE '9'
  @ 6,32 GET O2A PICTURE '!!'
  @ 6,35 GET F2A PICTURE '!!!!!!!!!!!!!!'
  @ 7,32 GET O2B PICTURE '!!'
  @ 7,35 GET F2B PICTURE '!!!!!!!!!!!!!!'
  @ 9,35 GET F3 PICTURE '@R 999-99-9999'
  @ 14,63 GET PRINT_OPT PICTURE '9' RANGE 1,2
```

\* Read query screen inputs and prepare to process them. \*

```
READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
```

\* If the user chooses to cancel the query, set the required \*  
\* flags to terminate all procedure loops. \*

```
IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE = .F.
  EXIT
ELSE
  STOP_LOOP = .F.
ENDIF
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
```

\* If the user wants to change their inputs, set DONE flag to \*  
\* false and repeat the current loop. \*

```
IF (DONE)
  @ 23, 0
  DONE = .F.
  LOOP
ELSE
  DONE = .T.
ENDIF
```

\* vvvvvvvvvvvvvvvvv #4. RELATIONAL OPERATOR CHECK vvvvvvvvvvvvvvvvv \*

```
GOOD_RO = .T.
TEMP_LOOP = .T.
DO WHILE (TEMP_LOOP)
```

```

        IF (O1A <> ' ')
            DO RO_CHK WITH O1A
            IF (.NOT. GOOD_RO)
                EXIT
            ENDIF
        ENDIF
        IF (O1B <> ' ')
            DO RO_CHK WITH O1B
            IF (.NOT. GOOD_RO)
                EXIT
            ENDIF
        ENDIF
        IF (O2A <> ' ')
            DO RO_CHK WITH O2A
            IF (.NOT. GOOD_RO)
                EXIT
            ENDIF
        ENDIF
        IF (O2B <> ' ')
            DO RO_CHK WITH O2B
            IF (.NOT. GOOD_RO)
                EXIT
            ENDIF
        ENDIF
        TEMP_LOOP = .F.
    ENDDO
    IF (.NOT. GOOD_RO)
        @ 23, 0
        ? CHR(7)
        M_CHOICE = .F.
        @ 23, 4 SAY 'INVALID RELATIONAL OPERATOR. WOULD YOU LIKE TO' ;
            + ' TRY AGAIN [Y/N]? ' GET M_CHOICE PICTURE 'Y'
        CLEAR TYPEAHEAD
        READ

        * Give the user the option of either returning to the      *
        * query input screen or terminating the query function.    *

        IF (M_CHOICE)
            @ 23, 0
            DONE = .F.
        ELSE
            STOP_LOOP = .T.
            EXIT
        ENDIF
    ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'.                                                  *

IF (STOP_LOOP)
    EXIT
ELSE

```

```

HDR1A = ''
HDR1B = ''
HDR2A = ''
HDR2B = ''
DATA1_S = ''
DATA1_L = ''
DATA2_S = ''
DATA2_L = ''
COL_HDRA = ' Max Min      Max '
COL_HDRB = ' WT  WT   10% RT '
COL_LIN = '| | | | |'
BLK_LINE = REPLICATE('_',80)
SEP_LINE = REPLICATE('_',80)
SQG_LINE = REPLICATE('~',80)
HDR1A = 'First              Last              Max  ';
      + '      Min              '
HDR1B = 'Name              Name              Heigh Weight Weight';
      + '      Weight              '
DATA1_S = "LEFT(F_NAME,14)+S2+LEFT(L_NAME,14)+S2+STR(HEIGHT,5,2)+S2";
      + "+STR(WEIGHT,6,2)+S2+STR(MAX_WGHT,6,2)+S2+STR(MIN_WGHT,6,2)";
      + "+S2+COL_LIN"
HDR2A = '              AS      Cat              Run  ';
      + '      Max              '+COL_LIN
HDR2B = '              Class Type      Age      Time  ';
      + '      Run Time '+COL_LIN
DATA2_S = "S26+STR(AS_CLASS,1)+S6+CAT_TYPE+S6+AGE+S5";
      + "+TRANSFORM(RUN_TIME,'@R 99:99')+S4";
      + "+TRANSFORM(STR(MAX_RT,4),'@R 99:99')+S2+COL_LIN"

IF (QO_SELECT = 'J')
  HDR1A = HDR1A + '      LOCAL'
  HDR1B = HDR1B + '      Street              City              Zip  ';
      + '      Phone'
  DATA1_L = "S2+LOCAL_STRT+S2+LEFT(LOCAL_CITY,15)+S2";
      + "+LEFT(LOCAL_ZIP,5)+S2+TRANSFORM(LOCAL_PHON,'@R 999-9999')";
  DATA2_L = "S2"
  SEP_LINE = SEP_LINE + REPLICATE('_',57)
  SQG_LINE = SQG_LINE + REPLICATE('~',57)
ENDIF

```

```
*  vvvvvvvvvvvvvvvvvvvv  #6.  BUILD FILTER STRING  vvvvvvvvvvvvvvvvvvvv  *
```

```

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
    FILT_STR = 'AS_CLASS' + O1A + F1A
ENDIF
IF (LEN(LTRIM(F1B)) > 0 .AND. (O1A <> O1B) .AND. (F1A <> F1B))
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.AS_CLASS' + O1B + F1B
    ELSE
        FILT_STR = 'AS_CLASS' + O1B + F1B
    ENDIF
ENDIF
ENDIF

```

```

IF (LEN(LTRIM(F2A)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O2A + '""' + F2A + '""'
  ELSE
    FILT_STR = 'L_NAME' + O2A + '""' + F2A + '""'
  ENDIF
ENDIF
IF (LEN(LTRIM(F2B)) > 0 .AND. (O2A <> O2B) .AND. (F2A <> F2B))
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.L_NAME' + O2B + '""' + F2B + '""'
  ELSE
    FILT_STR = 'L_NAME' + O2B + '""' + F2B + '""'
  ENDIF
ENDIF
IF (LEN(LTRIM(F3)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.SSAN =' + '""' + F3 + '""'
  ELSE
    FILT_STR = 'SSAN =' + '""' + F3 + '""'
  ENDIF
ENDIF
DONE = .T.

```

\* vvvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvvv \*

```

@ 23, 0
@ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
SELECT 1
IF (.NOT. FILE(M_NDX_F))
  INDEX ON &M_NDX_STR TO &M_NDX
ENDIF
SET INDEX TO &M_NDX
IF (LEN(FILT_STR) > 0)
  SET FILTER TO &FILT_STR
ENDIF
GOTO TOP
DO CASE

```

```

* If none of the database records meet all the input *
* constraints, give the user the option to try again *
* or to terminate the query. *

```

```

CASE (EOF())
DO ERR_NF
IF (M_CHOICE)
  DONE = .F.
  LOOP
ELSE
  EXIT
ENDIF

```

```

* If some database records meet the constraints, ini- *
* tialize the print environment and perform print loop *
* until all records are printed. *

```

```

CASE (.NOT. EOF())
  IF QO_SELECT <> 'H'
    SET PRINT ON
    SET DEVICE TO PRINT
    IF QO_SELECT = 'J'
      @ 0, 1 SAY CHR(27) + CHR(15)
    ELSE
      @ 0, 1 SAY CHR(27) + CHR(77)
    ENDIF
    MAX_LINES = 66
  ELSE
    MAX_LINES = 23
  ENDIF
  IF (QO_SELECT <> 'J')
    SPACER = SPACE(19)
  ELSE
    SPACER = SPACE(50)
  ENDIF
  CLEAR
  @ 0, 0 SAY SPACER + 'CADET WEIGHT AND AEROBIC STANDARDS';
    + ' REPORT'
  @ 1, 0
  FIRST_TIME = .T.
  DISP_LINE = 2

      * vvvvvvvvvvv #8. DATABASE RECORD LOOP vvvvvvvvvvv *

DO WHILE (.NOT. EOF())
  IF ((DISP_LINE > 0) .AND. (QO_SELECT <> 'H'))
    IF (.NOT. FIRST_TIME)
      EJECT
    ENDIF
  ENDIF
  IF (FIRST_TIME)
    FIRST_TIME = .F.
  ELSE
    DISP_LINE = 0
    CLEAR
  ENDIF

      * vvvvvvvvvvvvvvv #9. PAGING LOOP vvvvvvvvvvvvvvv *

DO WHILE ((DISP_LINE < MAX_LINES) .AND. (.NOT. EOF()))
  REC_NUM = RECNO()
  PRNT_FLAG = .F.
  VIOL_BAR = COL_LIN
  HGHT_SAV = HEIGHT
  SEX_SAV = SEX
  AGE_GROUP = '1'
  IF (INT(VAL(AGE))) >= 30)
    AGE_GROUP = '2'
  ENDIF
  SELECT 2
  SEEK HGHT_SAV
  MAX_WGHT = 0.00

```



```

MIN_WGHT = 0.00
IF (.NOT. EOF())
  IF (SEX_SAV = 'F')
    MAX_WGHT = MAX_WT_F
    MIN_WGHT = MIN_WT_F
  ELSE
    IF (SEX_SAV = 'M')
      MAX_WGHT = MAX_WT_M
      MIN_WGHT = MIN_WT_M
    ENDIF
  ENDIF
ENDIF
SELECT 3
SEEK AGE_GROUP
MAX_RT = 0000
IF (.NOT. EOF())
  IF (SEX_SAV = 'F')
    MAX_RT = MAX_RT_F
  ELSE
    IF (SEX_SAV = 'M')
      MAX_RT = MAX_RT_M
    ENDIF
  ENDIF
ENDIF
SELECT 1
GOTO REC_NUM
IF (WEIGHT > MAX_WGHT)
  PRNT_FLAG = .T.
  VIOL_BAR = STUFF(VIOL_BAR,3,1,'*')
ENDIF
IF (WEIGHT < MIN_WGHT)
  PRNT_FLAG = .T.
  VIOL_BAR = STUFF(VIOL_BAR,7,1,'*')
ENDIF
IF (WEIGHT > (MAX_WGHT*.90))
  PRNT_FLAG = .T.
  VIOL_BAR = STUFF(VIOL_BAR,11,1,'*')
ENDIF
IF (VAL(RUN_TIME) > MAX_RT)
  PRNT_FLAG = .T.
  VIOL_BAR = STUFF(VIOL_BAR,15,1,'*')
ENDIF
IF (PRINT_OPT = 1) .OR. (PRNT_FLAG)

  * If the number of print lines per cadet will *
  * not fit on one page, exit the loop and go to *
  * the next page. *

  IF ((MAX_LINES - DISP_LINE) < 7)
    EXIT
  ELSE
    IF (DISP_LINE <= 3)
      HDR1A = STUFF(HDR1A,64,17,COL_HDRA)
      HDR1B = STUFF(HDR1B,64,17,COL_HDRB)
      SEP_LINE = STUFF(SEP_LINE,64,17,REPLICATE('_',17))
    
```

```

ELSE
    HDR1A = STUFF(HDR1A,64,17,COL_LIN)
    HDR1B = STUFF(HDR1B,64,17,COL_LIN)
ENDIF
@ DISP_LINE, 0 SAY HDR1A
@ DISP_LINE + 1, 0 SAY HDR1B
*

IF (QO_SELECT <> 'H')
    @ DISP_LINE + 1, 0 SAY SEP_LINE
ENDIF
@ DISP_LINE + 2, 0 SAY &DATA1_S
*

IF (QO_SELECT = 'J')
    @ DISP_LINE + 2, 80 SAY &DATA1_L
ENDIF
BLK_LINE = STUFF(BLK_LINE,64,17,VIOL_BAR)
@ DISP_LINE + 3, 0 SAY BLK_LINE
@ DISP_LINE + 4, 0 SAY HDR2A
@ DISP_LINE + 5, 0 SAY HDR2B
*

IF (QO_SELECT <> 'H')
    SEP_LINE = STUFF(SEP_LINE,64,17,COL_LIN)
    SEP_LINE = STUFF(SEP_LINE,1,26,S26)
    @ DISP_LINE + 5, 0 SAY SEP_LINE
SEP_LINE = STUFF(SEP_LINE,1,26,REPLICATE('_',26))
ENDIF
@ DISP_LINE + 6, 0 SAY &DATA2_S
*

IF (QO_SELECT = 'J')
    @ DISP_LINE + 6, 80 SAY &DATA2_L
ENDIF
SQG_LINE = STUFF(SQG_LINE,64,17,COL_LIN)
@ DISP_LINE + 7, 0 SAY SQG_LINE
DISP_LINE = DISP_LINE + 8
ENDIF
ENDIF

* Issue dBASE III PLUS command to go to the *
* next record which meets the input constraints.*

SKIP
ENDDO

* If the output media is the screen, issue the user*
* paging prompt. *

IF (QO_SELECT = 'H')
    @ 23, 0 SAY 'PRESS ANY KEY TO CONTINUE'
    CLEAR TYPEAHEAD
    WAIT ''
ENDIF
ENDDO
IF (QO_SELECT <> 'H')
    @ DISP_LINE + 1, 0 SAY CHR(10)
    EJECT

```

```

        IF (QO_SELECT = 'J')
            @ 0, 1 SAY CHR(18)
        ELSE
            @ 0, 1 SAY CHR(27) + CHR(80)
        ENDIF
        SET PRINT OFF
    ENDIF
    SET DEVICE TO SCREEN
    SET FILTER TO

ENDCASE
ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *
* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
    DO RCIS_HDR
    DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

SELECT 3
USE
SELECT 2
USE
SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```



```

* Read query screen inputs and prepare to process them.  *

READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

* If the user chooses to cancel the query, set the required  *
* flags to terminate all procedure loops.                    *

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE  = .F.
  EXIT
ELSE
  STOP_LOOP = .F.
ENDIF
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

* If the user wants to change their inputs, set DONE flag to  *
* false and repeat the current loop.                          *

IF (DONE)
  @ 23, 0
  DONE = .F.
  LOOP
ELSE
  DONE = .T.
ENDIF
ENDDO

* Check to see if query termination condition has been previously *
* set to 'true'.                                                  *

IF (STOP_LOOP)
  EXIT

* vvvvvvvvvvvvvv #5. BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvvv *

ELSE
  HDR1A = ''
  HDR1B = ''
  HDR2A = ''
  HDR2B = ''
  HDR3A = ''
  HDR3B = ''
  HDR4A = ''
  HDR4B = ''

```

```

HDR5A = ''
HDR5B = ''
HDR6A = ''
HDR6B = ''
DATA1_S = ''
DATA1_L = ''
DATA2_S = ''
DATA2_L = ''
DATA3_S = ''
DATA4_S = ''
DATA5_S = ''
DATA5_L = ''
DATA6_S = ''
DATA6_L = ''
HDR1A = 'First           Middle   Last           ';
      + '           Birth           '
HDR1B = 'Name           Name       Name           SSAN           ';
      + '   Matric Date       Age Sex'
DATA1_S = "LEFT(F_NAME,14)+S2+LEFT(M_NAME,7)+S2+LEFT(L_NAME,14)+S2";
      + "'+TRANSFORM(SSAN,'@R 999-99-9999')+S2+MATRIC+S2+DLOC(BIRTHDATE)";
      + "'S3+AGE+S3+SEX"

```

\*

```

HDR2A = 'AS  AS Class  DC    FY    FT    FT           Pil           ';
      + '   Corps'
HDR2B = 'Yr    Rank    Rtng Rtng Rating Cmp ALTU Lics Work';
      + '   Auxiliaries'
DATA2A = "''+STR(AS_CLASS,1)+S3+STR(AS_RNK_POS,3)+'/'+CLAS_NUM+S3";
      + "'+STR(DC_RTNG,1)+S5+STR(FY_RTNG,2)+S3+STR(FT_RTNG,6,2)+S3";
      + "'+FTC+S4+ALT+S5"
DATA2B = "'+PLS+S5+WRK+S4+TRANSFORM(CORPS_AUX,'@R !!!!!!!!!!!!!!!')";
DATA2_S = DATA2A + DATA2B

```

\*

```

HDR3A = 'Cat  Purs  4-Yr  Pri  Waiv Form 48  Semester           ';
      + '   FSP'
HDR3B = 'Type Cond Cad.  Serv Req  Date           Intrview Race';
      + '   Date'
DATA3_S = "''+CAT_TYPE+S5+PC_STATUS+S5+FYC+S5+PRS+S5+WRQ+S4";
      + "'+DLOC(FORM_48)+S2+DLOC(SEM_INTRVW)+S3+RACE+S4+DLOC(FSP_DATE)";

```

\*

```

HDR4A = '           Weigh    Run    Run           Phys Phys           ';
      + '   Grad      Comm'
HDR4B = 'Height Weight Date           Time    Date           Cat    Date           ';
      + '   Date      Date'
DATA4_S = "''+STR(HEIGHT,5,2)+S2+STR(WEIGHT,6,2)+S2+DLOC(WEIGH_DATE)";
      + "'+S2+TRANSFORM(RUN_TIME,'@R 99:99')+S2+DLOC(RUN_DATE)+S3+PHY_CAT";
      + "'+S4+DLOC(PHY_DATE)+S2+DLOC(GRAD_DATE)+S2+DLOC(COM_DATE)+S3"

```

\*

```

HDR5A = '           Schl  Schl Exp  GPA           SAT           ';
      + '   ACT'
HDR5B = 'Major Type Date           Cum    Sem    Cum    Math Verb';
      + '   Cum Math Engl  NSci  SSci'
DATA5A = "''+MAJOR+S3+STR(SCHLR_TYPE,3,1)+S2+DLOC(SCHLR_DATE)+S2";
      + "'+STR(CUM_GPA,4,2)+S2+STR(SEM_GPA,4,2)+S2+STR(SAT_CUM,4)+S3";
      + "'+STR(SAT_MATH,3)+S3+STR(SAT_VERB,3)+S3+STR(ACT_CUM,2)+S3";
DATA5B = "'+STR(ACT_MATH,2)+S4+STR(ACT_ENGL,2)+S4+STR(ACT_NSCI,2)+S4";

```

```

      + "+STR(ACT_SSCI,2)"
DATA5_S = DATA5A + DATA5B

*
HDR6A = 'AFOQT                      AFOQT      Min Req      '
HDR6B = 'Quan  Verb  Pil  Nav  AcAp  Date      Math  Engl  Frln'
DATA6_S = "' '+STR(AFOQT_QUAN,2)+S4+STR(AFOQT_VERB,2)+S4";
      + "+STR(AFOQT_PLT,2)+S3+STR(AFOQT_NAV,2)+S3+STR(AFOQT_AA,2)";
      + "+S3+DLOC(AFOQT_DATE)+S3+MRM+S5+MRE+S5+MRF+S2"

*
SEP_LINE = REPLICATE('-',80)
SQG_LINE = REPLICATE('~',80)

*
IF (QO_SELECT = 'J')
  HDR1A = HDR1A + '  LOCAL'
  HDR1B = HDR1B + '  Street                      City              Zip  ';
      + '  Phone'
  DATA1_L = "S2+LOCAL_STRT+S2+LEFT(LOCAL_CITY,15)+S2";
      + "+LEFT(LOCAL_ZIP,5)+S2+TRANSFORM(LOCAL_PHON,'@R 999-9999')";

*
  HDR2A = HDR2A + '  Corps'
  HDR2B = HDR2B + '  Position'
  DATA2_L = "S2+CORPS_POS"

*
  HDR3A = HDR3A + '  PERMANENT'
  HDR3B = HDR3B + '  Street                      City              ';
      + '  ST Zip Phone'
  DATA3_L = "S2+LEFT(PERM_STRT,19)+S2+LEFT(PERM_CITY,19)+S2";
      + "+PERM_STAT+S2+TRANSFORM(PERM_ZIP,'@R 99999-NNNN')+S2";
      + "+TRANSFORM(PERM_PHON,'@R (999)999-9999')";

*
  HDR6A = HDR6A + '  Significant'
  HDR6B = HDR6B + '  Information'
  DATA6_L = "S2+OTHER_INFO"

*
  SEP_LINE = SEP_LINE + REPLICATE('-',57)
  SQG_LINE = SQG_LINE + REPLICATE('~',57)
ENDIF

* vvvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvv *

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
  FILT_STR = 'F_NAME = ' + "'" + F1A + "'"
ENDIF
IF (LEN(LTRIM(F1B)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.M_NAME = ' + "'" + F1B + "'"
  ELSE
    FILT_STR = 'M_NAME = ' + "'" + F1B + "'"
  ENDIF
ENDIF
IF (LEN(LTRIM(F1C)) > 0)
  IF (LEN(FILT_STR) > 0)
    FILT_STR = FILT_STR + '.AND.I_NAME = ' + "'" + F1C + "'"
  ELSE
    FILT_STR = 'I_NAME = ' + "'" + F1C + "'"
  ENDIF
ENDIF

```

```

        FILT_STR = 'L_NAME =' + "" + F1C + ""
    ENDIF
ENDIF
IF (LEN(LTRIM(F2)) > 0)
    IF (LEN(FILT_STR) > 0)
        FILT_STR = FILT_STR + '.AND.SSAN =' + "" + F2 + ""
    ELSE
        FILT_STR = 'SSAN =' + "" + F2 + ""
    ENDIF
ENDIF
ENDIF
DONE = .T.

* vvvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvvv *

IF (LEN(FILT_STR) > 0)
    @ 23, 0
    @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
    SELECT 1
    IF (.NOT. FILE(M_NDX_F))
        INDEX ON &M_NDX_STR TO &M_NDX
    ENDIF
    SET INDEX TO &M_NDX
    IF (LEN(LTRIM(F2)) = 0)
        COUNT FOR &FILT_STR TO REC_CNT
        IF (REC_CNT > 1)
            @ 23, 0
            ? CHR(7)
            @ 23, 0 SAY 'NAME ASSIGNED TO MORE THAN ONE RECORD (ENTER';
                + ' SSAN). PRESS ANY KEY & TRY AGAIN.'
            WAIT ''
            DONE = .F.
            LOOP
        ENDIF
    ENDIF
    SET FILTER TO &FILT_STR
    GOTO TOP
DO CASE

    * If none of the database records meet all the input *
    * constraints, give the user the option to try again *
    * or to terminate the query. *

    CASE (EOF())
        DO ERR_NF
        IF (M_CHOICE)
            DONE = .F.
            LOOP
        ELSE
            EXIT
        ENDIF

    * If some database records meet the constraints, ini- *
    * tialize the print environment and perform print func- *
    * tion until all data is printed. *
```



```

CASE (.NOT. EOF())
  REC_NUM = RECNO()
  IF QO_SELECT <> 'H'
    SET PRINT ON
    SET DEVICE TO PRINT
    IF QO_SELECT = 'J'
      @ 0, 1 SAY CHR(27) + CHR(15)
    ELSE
      @ 0, 1 SAY CHR(27) + CHR(77)
    ENDIF
  ENDIF
  IF (QO_SELECT <> 'J')
    SPACER = SPACE(27)
  ELSE
    SPACER = SPACE(59)
  ENDIF
  CLEAR
  DISP_LINE = 0

  IF (QO_SELECT <> 'H')
    DISP_LINE = 5
  ENDIF
  @ DISP_LINE, 0 SAY SPACER + 'INDIVIDUAL CADET REPORT'

  IF (QO_SELECT <> 'H')
    DISP_LINE = DISP_LINE + 1
  ENDIF
  @ DISP_LINE + 2, 0 SAY HDR1A
  @ DISP_LINE + 3, 0 SAY HDR1B

  IF (QO_SELECT <> 'H')
    @ DISP_LINE + 3, 0 SAY SEP_LINE
  ENDIF
  @ DISP_LINE + 4, 0 SAY &DATA1_S

  IF (QO_SELECT = 'J')
    @ DISP_LINE + 4, 80 SAY &DATA1_L
  ENDIF

  IF (QO_SELECT <> 'H')
    DISP_LINE = DISP_LINE + 1
  ENDIF
  @ DISP_LINE + 6, 0 SAY HDR2A
  @ DISP_LINE + 7, 0 SAY HDR2B

  IF (QO_SELECT <> 'H')
    @ DISP_LINE + 7, 0 SAY SEP_LINE
  ENDIF
  FTC = 'N'
  ALT = 'N'
  PLS = 'N'
  WRK = 'N'
  IF FT_COMP
    FTC = 'Y'
  ENDIF

```

```

IF ALTU
  ALT = 'Y'
ENDIF
IF PLT_LICENS
  PLS = 'Y'
ENDIF
IF WORK
  WRK = 'Y'
ENDIF
CLAS_VAL = AS_CLASS
SELECT 2
SEEK CLAS_VAL
IF (.NOT. EOF())
  CLAS_NUM = STR(AS_CL_TOT,3)
ELSE
  CLAS_NUM = ' ? '
ENDIF
SELECT 1
GOTO REC_NUM
@ DISP_LINE + 8, 0 SAY &DATA2_S
*

IF (QO_SELECT = 'J')
  @ DISP_LINE + 8, 80 SAY &DATA2_L
ENDIF
*

IF (QO_SELECT <> 'H')
  DISP_LINE = DISP_LINE + 1
ENDIF
@ DISP_LINE + 10, 0 SAY HDR3A
@ DISP_LINE + 11, 0 SAY HDR3B
*

IF (QO_SELECT <> 'H')
  @ DISP_LINE + 11, 0 SAY SEP_LINE
ENDIF
FYC = 'N'
PRS = 'N'
WRQ = 'N'
IF FOUR_YR
  FYC = 'Y'
ENDIF
IF PRIOR_SVC
  PRS = 'Y'
ENDIF
IF WAIVER_REQ
  WRQ = 'Y'
ENDIF
@ DISP_LINE + 12, 0 SAY &DATA3_S
*

IF (QO_SELECT = 'J')
  @ DISP_LINE + 12, 64 SAY &DATA3_L
ENDIF
*

IF (QO_SELECT <> 'H')
  DISP_LINE = DISP_LINE + 1
ENDIF

```

```

        @ DISP_LINE + 14, 0 SAY HDR4A
        @ DISP_LINE + 15, 0 SAY HDR4B
*
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 15, 0 SAY SEP_LINE
        ENDIF
        @ DISP_LINE + 16, 0 SAY &DATA4_S
*
        IF (QO_SELECT <> 'H')
            DISP_LINE = DISP_LINE + 1
        ENDIF
        @ DISP_LINE + 18, 0 SAY HDR5A
        @ DISP_LINE + 19, 0 SAY HDR5B
*
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 19, 0 SAY SEP_LINE
        ENDIF
        DL = DISP_LINE + 20

* The position of the following line is critical for it to print properly. *
* The string variable is so long that DOS will not accept it unless it is *
* <= 256 characters when combined with the other commands on the same line.*

*****
@ DL, 0 SAY &DATA5_S
*****
        IF (QO_SELECT <> 'H')
            DISP_LINE = DISP_LINE + 1
        ENDIF
        @ DISP_LINE + 22, 0 SAY HDR6A
        @ DISP_LINE + 23, 0 SAY HDR6B
*
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 23, 0 SAY SEP_LINE
        ENDIF
        MRM = 'N'
        MRE = 'N'
        MRF = 'N'
        IF M_R_MATH
            MRM = 'Y'
        ENDIF
        IF M_R_ENGL
            MRE = 'Y'
        ENDIF
        IF M_R_FLAN
            MRF = 'Y'
        ENDIF
        @ DISP_LINE + 24, 0 SAY &DATA6_S
*
        IF (QO_SELECT = 'J')
            @ DISP_LINE + 24, 54 SAY &DATA6_L
        ENDIF
*
        IF (QO_SELECT <> 'H')
            @ DISP_LINE + 26, 0 SAY SQG_LINE

```

```

ENDIF

* If the output media is the screen, issue the user*
* paging prompt. *

IF (QO_SELECT = 'H')
  @ 0,52 SAY '(Press any key to continue)'
  CLEAR TYPEAHEAD
  WAIT ''
ENDIF

*

IF (QO_SELECT <> 'H')
  @ DISP_LINE + 27, 0 SAY CHR(10)
  EJECT
  IF (QO_SELECT = 'J')
    @ 0, 1 SAY CHR(18)
  ELSE
    @ 0, 1 SAY CHR(27) + CHR(80)
  ENDIF
  SET PRINT OFF
ENDIF
SET DEVICE TO SCREEN
SET FILTER TO

ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure). *

ELSE
  @ 23, 0
  ? CHR(7)
  @ 23, 4 SAY 'PLEASE ENTER DATA. PRESS ANY KEY TO CONTINUE.'
  CLEAR TYPEAHEAD
  WAIT ''
  @ 23, 0
  DONE = .F.
ENDIF

ENDIF
ENDDO
CLEAR

* If the user has not previously entered a response to terminate the *
* query (M_CHOICE would be "false"), then give them the opportunity *
* to do another query or terminate the function. *

IF (M_CHOICE)
  DO RCIS_HDR
  DO M_PROMPT
ENDIF
ENDDO

* Close the database files used in this query. *

```

```
SELECT 2
USE
SELECT 1
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN
```

```

*-----*
*                                PAYI_QRY                                *
*-----*
*
* SUMMARY:
*   The PAYI_QRY procedure provides the interface for the user to per- *
*   form queries on all the data contained in the associated Pay re-   *
*   cords of an individual cadet. All data is displayed on one screen.*
*-----*

```

# ``` PROCEDURE PAYI_QRY ```

```

*
PRIVATE FYC
PRIVATE PRS
PRIVATE WRQ
PRIVATE PLS
PRIVATE SPACER
*
ON ERROR DO DB3_Q_ERR WITH ERROR(), MESSAGE()
CLEAR
M_CHOICE = .T.

* vvvvvvvvvvvvvvvvvvvvvvvv #1. MAIN OUTER LOOP vvvvvvvvvvvvvvvvvvvvvvvv *

DO WHILE (M_CHOICE)

    * Initialize operator and constraint fields. *

    DONE = .F.
    F1A = ' '
    F1B = ' '
    F1C = ' '
    F2 = ' '

    * vvvvvvvvvvvvvvvvvvvvvvvv #2. INTERMEDIATE SCREEN LOOP vvvvvvvvvvvvvvvvvvvvvvvv *

    DO WHILE (.NOT. DONE)
        CLEAR
        @ 5, 0 TO 15,79
        @ 5,26 SAY ' INDIVIDUAL CADET PAY QUERY '
        @ 7,24 SAY 'Enter Name or Social Security #'
        @ 9,27 SAY 'First Name'
        @ 10,26 SAY 'Middle Name'
        @ 11,28 SAY 'Last Name'
        @ 13,33 SAY 'SSAN'

        * vvvvvvvvvvvvvvvvvvvvvvvv #3. INTERMEDIATE INPUT LOOP vvvvvvvvvvvvvvvvvvvvvvvv *

        DO WHILE (.NOT. DONE)
            @ 9,38 GET F1A PICTURE '!!!!!!!!!!!!!!'
            @ 10,38 GET F1B PICTURE '!!!!!!!!!!!!!!'
            @ 11,38 GET F1C PICTURE '!!!!!!!!!!!!!!'
            @ 13,38 GET F2 PICTURE '@R 999-99-9999'

```

```

*   Read query screen inputs and prepare to process them.   *

READ
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO CANCEL THIS QUERY [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

*   If the user chooses to cancel the query, set the required *
*   flags to terminate all procedure loops.                  *

IF (DONE)
  STOP_LOOP = .T.
  M_CHOICE  = .F.
  EXIT
ELSE
  STOP_LOOP = .F.
ENDIF
@ 23, 0
@ 23,19 SAY ;
      "DO YOU WANT TO MAKE ANY CHANGES [Y/N]? " GET DONE PICTURE 'Y'
CLEAR TYPEAHEAD
READ

*   If the user wants to change their inputs, set DONE flag to *
*   false and repeat the current loop.                          *

IF (DONE)
  @ 23, 0
  DONE = .F.
  LOOP
ELSE
  DONE = .T.
ENDIF
ENDDO

*   Check to see if query termination condition has been previously *
*   set to 'true'.                                                  *

IF (STOP_LOOP)
  EXIT
ELSE

*   vvvvvvvvvvvvvv #5.  BUILD QUERY OUTPUT FORMAT vvvvvvvvvvvvvv *

  HDR1A = ''
  HDR1B = ''
  HDR2A = ''
  HDR2B = ''
  DATA1_S = ''
  DATA1_L = ''
  DATA2_S = ''
  DATA2_L = ''

```

```

HDR1A = 'First           Middle   Last           ';
+ '           AS      Cat   Schl'
HDR1B = 'Name           Name       Name           SSAN           ';
+ 'Matric Class Type Type'
DATA1_S = "LEFT(FN,14)+S2+LEFT(MN,7)+S2+LEFT(LN,14)+S2";
+ "+TRANSFORM(F2,'@R 999-99-9999')+S2+MT+S3+STR(ASC,1)+S6";
+ "+CT+S5+STR(ST,3,1)"
HDR2A = ' Pay      Start      Stop      Res           Book      FT ';
+ ' ATP      FSP      Num      Cum'
HDR2B = 'Period Pay Date Pay Date Stat Tuition Fees Days';
+ ' Days Days Days Days'
DATA2_S = "S2+STR(REC_NUM,2)+S4+DTC(PAY_DATE1)+S2+DTC(PAY_DATE2)+S3";
+ "+RES_STATUS+S4+STR(TUITION,7,2)+S3+STR(BOOK_FEES,6,2)+S3+STR(FT_DAYS,2)+S4";
+ "+STR(ATP_DAYS,2)+S4+STR(FSP_DAYS,2)+S4+STR(SUB_DAYS,3)+S2+STR(TOT_DAYS,4)"
*
DATA_TOTS = "'(Column Totals)-->           '+STR(TOT_TUIT,8,2)+S2";
+ "+STR(TOT_BKFE,7,2)+S3+STR(TOT_FTDY,2)+S4+STR(TOT_ATPD,2)";
+ "+S4+STR(TOT_FSPD,2)"
SEP_LINE = REPLICATE('-',80)
SQG_LINE = REPLICATE('~',80)
*
IF (QO_SELECT = 'J')
DATA1_L = "S2"
DATA2_L = "S2"
SEP_LINE = SEP_LINE + REPLICATE('-',57)
SQG_LINE = SQG_LINE + REPLICATE('~',57)
ENDIF

* vvvvvvvvvvvvvvvvv #6. BUILD FILTER STRING vvvvvvvvvvvvvvvvv *

FILT_STR = ''
IF (LEN(LTRIM(F1A)) > 0)
FILT_STR = 'F_NAME =' + "'" + F1A + "'"
ENDIF
IF (LEN(LTRIM(F1B)) > 0)
IF (LEN(FILT_STR) > 0)
FILT_STR = FILT_STR + '.AND.M_NAME =' + "'" + F1B + "'"
ELSE
FILT_STR = 'M_NAME =' + "'" + F1B + "'"
ENDIF
ENDIF
IF (LEN(LTRIM(F1C)) > 0)
IF (LEN(FILT_STR) > 0)
FILT_STR = FILT_STR + '.AND.L_NAME =' + "'" + F1C + "'"
ELSE
FILT_STR = 'L_NAME =' + "'" + F1C + "'"
ENDIF
ENDIF
IF (LEN(LTRIM(F2)) > 0)
IF (LEN(FILT_STR) > 0)
FILT_STR = FILT_STR + '.AND.SSAN =' + "'" + F2 + "'"
ELSE
FILT_STR = 'SSAN =' + "'" + F2 + "'"
ENDIF
ENDIF
ENDIF

```



DONE = .T.

\* vvvvvvvvvv #7. ACCESS DATABASE & DIRECT OUTPUT vvvvvvvvvv \*

```
IF (LEN(FILT_STR) > 0)
  @ 23, 0
  @ 23,14 SAY 'SEARCHING DATABASE FILES FOR CORRESPONDING RECORD(S)'
  SELECT 1
  IF (.NOT. FILE(M_NDX_F))
    INDEX ON &M_NDX_STR TO &M_NDX
  ENDIF
  SET INDEX TO &M_NDX
  IF (LEN(LTRIM(F2)) = 0)
    COUNT FOR &FILT_STR TO REC_CNT
    IF (REC_CNT > 1)
      @ 23, 0
      ? CHR(7)
      @ 23, 0 SAY 'NAME ASSIGNED TO MORE THAN ONE RECORD (ENTER';
        + ' SSAN). PRESS ANY KEY & TRY AGAIN.'
      WAIT ''
      DONE = .F.
      LOOP
    ENDIF
  ENDIF
  SET FILTER TO &FILT_STR
  GOTO TOP
DO CASE
```

```
* If no Master record exists for the input key con- *
* straints, give the user the option to try again or *
* to terminate the query. *
*
```

```
CASE EOF()
  DO ERR_NF
  IF (M_CHOICE)
    DONE = .F.
    LOOP
  ELSE
    EXIT
  ENDIF
```

```
* If some database records meet the constraints, ini- *
* tialize the print environment and perform print loop *
* until all records are printed. *
```

```
CASE .NOT. EOF()
  F2 = SSAN
  FN = F_NAME
  MN = M_NAME
  LN = L_NAME
  MT = MATRIC
  ASC = AS_CLASS
  CT = CAT_TYPE
  ST = SCHLR_TYPE
  SELECT 2
```

```

IF (.NOT. FILE(P_NDX_F))
  INDEX ON &P_NDX_STR TO &P_NDX
ENDIF
SET INDEX TO &P_NDX
SET FILTER TO SSAN = F2
SEEK F2
DO CASE

```

```

* If none of the database records meet all the *
* input constraints, give the user the option to*
* try again or to terminate the query.      *

```

```

CASE EOF()
  DO ERR_NF
  IF (M_CHOICE)
    DONE = .F.
    LOOP
  ELSE
    EXIT
  ENDIF

```

```

* If some database records meet the constraints, *
* initialize the print environment and perform *
* print loop until all records are printed.      *

```

```

CASE .NOT. EOF()
  IF QO_SELECT <> 'H'
    SET PRINT ON
    SET DEVICE TO PRINT
    IF QO_SELECT = 'J'
      @ 0, 1 SAY CHR(27) + CHR(15)
    ELSE
      @ 0, 1 SAY CHR(27) + CHR(77)
    ENDIF
  ENDIF
  IF (QO_SELECT <> 'J')
    SPACER = SPACE(23)
  ELSE
    SPACER = SPACE(57)
  ENDIF
  CLEAR
  DISP_LINE = 0

  IF (QO_SELECT <> 'H')
    DISP_LINE = 5
  ENDIF
  @ DISP_LINE, 0 SAY ;
  SPACER + 'INDIVIDUAL CADET PAY REPORT'

  IF (QO_SELECT <> 'H')
    DISP_LINE = DISP_LINE + 1
  ENDIF
  @ DISP_LINE + 2, 0 SAY HDR1A
  @ DISP_LINE + 3, 0 SAY HDR1B

```

```

IF (QO_SELECT <> 'H')
  @ DISP_LINE + 3, 0 SAY SEP_LINE
ENDIF
@ DISP_LINE + 4, 0 SAY &DATA1_S
*

IF (QO_SELECT = 'J')
  @ DISP_LINE + 4, 80 SAY &DATA1_L
ENDIF
*

IF (QO_SELECT <> 'H')
  DISP_LINE = DISP_LINE + 1
ENDIF
@ DISP_LINE + 6, 0 SAY HDR2A
@ DISP_LINE + 7, 0 SAY HDR2B
*

IF (QO_SELECT <> 'H')
  @ DISP_LINE + 7, 0 SAY SEP_LINE
ENDIF
DISP_LINE = DISP_LINE + 8
REC_NUM = 1
TOT_DAYS = 0
TOT_TUIT = 0
TOT_BKFE = 0
TOT_FTDY = 0
TOT_ATPD = 0
TOT_FSPD = 0

* vvvvv #8. DATABASE RECORD LOOP vvvvv *

DO WHILE (REC_NUM <= 16) .AND. (.NOT. EOF())
  SUB_DAYS = (PAY_DATE2-PAY_DATE1)+1-FT_DAYS;
             -FSP_DAYS-ATP_DAYS
  TOT_DAYS = TOT_DAYS + SUB_DAYS
  TOT_TUIT = TOT_TUIT + TUITION
  TOT_BKFE = TOT_BKFE + BOOK_FEES
  TOT_FTDY = TOT_FTDY + FT_DAYS
  TOT_ATPD = TOT_ATPD + ATP_DAYS
  TOT_FSPD = TOT_FSPD + FSP_DAYS
  DL = DISP_LINE

* The position of the following line is critical for it to print properly. *
* The string variable is so long that DOS will not accept it unless it is *
* <= 256 characters when combined with the other commands on the same line.*

*****
@ DL, 0 SAY &DATA2_S
*****

IF (QO_SELECT = 'J')
  @ DISP_LINE, 80 SAY &DATA2_L
ENDIF
DISP_LINE = DISP_LINE + 1
*

IF (QO_SELECT <> 'H')
  DISP_LINE = DISP_LINE + 1
ENDIF

```

```

REC_NUM = REC_NUM + 1

* Issue dBASE III PLUS command to go to *
* the next record which meets the input *
* constraints. *

SKIP
ENDDO

*
IF (QO_SELECT <> 'H')
  @ DISP_LINE - 1, 0 SAY SEP_LINE
ENDIF
@ DISP_LINE, 0 SAY &DATA_TOTS

*
IF (QO_SELECT <> 'H')
  @ DISP_LINE + 2, 0 SAY SQG_LINE
ENDIF

* If the output media is the screen, issue *
* the user paging prompt. *

IF (QO_SELECT = 'H')
  @ 0,52 SAY '(Press any key to continue)'
  CLEAR TYPEAHEAD
  WAIT ''
ENDIF

*
IF (QO_SELECT <> 'H')
  @ DISP_LINE + 27, 0 SAY CHR(10)
  EJECT
  IF (QO_SELECT = 'J')
    @ 0, 1 SAY CHR(18)
  ELSE
    @ 0, 1 SAY CHR(27) + CHR(60)
  ENDIF
  SET PRINT OFF
ENDIF
SET DEVICE TO SCREEN
SET FILTER TO

ENDCASE

ENDCASE

* If the user fails to enter any data in the input fields, *
* issue a prompt for them to please enter data (if they had *
* intended to cancel the query, they should not have gotten *
* this far in the procedure). *

ELSE
  @ 23, 0
  ? CHR(7)
  @ 23, 4 SAY 'PLEASE ENTER DATA. PRESS ANY KEY TO CONTINUE.'
  CLEAR TYPEAHEAD
  WAIT ''
  @ 23, 0
  DONE = .F.

```

```

        ENDIF
    ENDIF
ENDDO
CLEAR

*   If the user has not previously entered a response to terminate the *
*   query (M_CHOICE would be "false"), then give them the opportunity *
*   to do another query or terminate the function.                       *

    IF (M_CHOICE)
        DO RCIS_HDR
        DO M_PROMPT
    ENDIF
ENDDO

*   Close the database files used in this query.  *

SELECT 1
USE
SELECT 2
USE
F_PARA = STUFF(F_PARA,1,1,'C')
ON ERROR
*
RETURN

```

```

*-----*
*                                HELP_SCRN                                *
*-----*
*
* SUMMARY:
*      The HELP_SCRN procedure builds a help menu at the bottom of each
*      query input screen which provides an example of how to enter query
*      requests.
*
* INVOKING PROCEDURES:
*
*      Procedure Name      Location
*      -----
*      WPSS_QRY            RCIS_P3.PRG
*      SCHA_QRY            RCIS_P3.PRG
*      DCFY_QRY            RCIS_P3.PRG
*      CLAS_QRY            RCIS_P3.PRG
*      HRAX_QRY            RCIS_P3.PRG
*      CGDT_QRY            RCIS_P3.PRG
*      SEDT_QRY            RCIS_P3.PRG
*      WTAR_QRY            RCIS_P3.PRG
*      INDV_QRY            RCIS_P3.PRG
*      PAYI_QRY            RCIS_P3.PRG
*-----*

```

# PROCEDURE HELP\_SCRN

```

*
HLP_O1 = '>='
HLP_O2 = '<'
HLP_V1 = 'ANDERSON'
HLP_V2 = 'SMITH'
@ 17,11 SAY "Query Item      Operators[<,>,<=>,<=>,<=>]      Query Values"
@ 18,11 TO 18,79
@ 19, 0 SAY "  EXAMPLE      Last Name      * Absence of Operator"
@ 20,33 SAY "field defaults to '='
@ 19,26 GET HLP_O1
@ 19,59 GET HLP_V1
@ 20,26 GET HLP_O2
@ 20,59 GET HLP_V2
@ 16, 0 TO 21,10
@ 16,10 TO 21,79
CLEAR GETS
*
RETURN

```

```

*-----*
*                                ERR_NF                                *
*-----*
*
* SUMMARY:
*      The ERR_NF procedure displays an error message informing the user
*      that a record with the requested key value doesn't exist and then
*      accepts a continuation option.
*
* INVOKING PROCEDURES:
*
*      Procedure Name      Location
*      -----
*      WPSS_QRY            RCIS_P3.PRG
*      SCHA_QRY            RCIS_P3.PRG
*      DCFY_QRY            RCIS_P3.PRG
*      CLAS_QRY            RCIS_P3.PRG
*      HRAX_QRY            RCIS_P3.PRG
*      CGDT_QRY            RCIS_P3.PRG
*      SEDT_QRY            RCIS_P3.PRG
*      WTAR_QRY            RCIS_P3.PRG
*      INDV_QRY            RCIS_P3.PRG
*      PAYI_QRY            RCIS_P3.PRG
*
*-----*

```

# PROCEDURE ERR\_NF

```

*
@ 23, 0
? CHR(7)
M_CHOICE = .T.
@ 23,11 SAY 'NO RECORD(S) FOUND. DO YOU WANT TO TRY AGAIN [Y/N]? ';
      GET M_CHOICE PICTURE 'Y'
CLEAR TYPEAHEAD
READ
@ 21, 0
@ 23, 0
IF .NOT. M_CHOICE
  @ 21,33 SAY 'CLOSING FILES'
  @ 24, 0
ENDIF
RETURN

```

```

*-----*
*                               RCIS_HDR                               *
*-----*
*
* SUMMARY:
*       The RCIS_HDR procedure redisplay the selected mode by repainting
*       the pop-up menus.
*
* INVOKING PROCEDURES:
*
*       Procedure Name      Location
*       -----
*       WPSS_QRY            RCIS_P3.PRG
*       SCHA_QRY            RCIS_P3.PRG
*       DCFY_QRY            RCIS_P3.PRG
*       CLAS_QRY            RCIS_P3.PRG
*       HRAX_QRY            RCIS_P3.PRG
*       CGDT_QRY            RCIS_P3.PRG
*       SEDT_QRY            RCIS_P3.PRG
*       WTAR_QRY            RCIS_P3.PRG
*       INDV_QRY            RCIS_P3.PRG
*       PAYI_QRY            RCIS_P3.PRG
*-----*

```

PROCEDURE RCIS\_HDR

```

*
  CLEAR
  @ 1, 0 TO 3,79
  @ 2,22 SAY 'ROTC CADET INFORMATION SYSTEM (RCIS)'
  CALL MENU WITH F_PARA
  CALL MENU WITH G_PARA
  IF F_SELECT = 'M'
    CALL MENU WITH QS_PARA
    CALL MENU WITH QO_PARA
  ELSE
    IF (F_SELECT <> 'L')
      CALL MENU WITH R_PARA
    ENDIF
  ENDIF
  @ 24, 0
*
  RETURN

```



```

*-----*
*                               M_PROMPT                               *
*-----*
*
* SUMMARY:                                                              *
*     The M_PROMPT procedure displays a continuation message and accepts *
*     the user option.                                                 *
*
* INVOKING PROCEDURES:                                                 *
*
*                               Procedure Name                          Location
*                               -----
*                               WPSS_QRY                               RCIS_P3.PRG
*                               SCHA_QRY                               RCIS_P3.PRG
*                               DCFY_QRY                               RCIS_P3.PRG
*                               CLAS_QRY                               RCIS_P3.PRG
*                               HRAX_QRY                               RCIS_P3.PRG
*                               CGDT_QRY                               RCIS_P3.PRG
*                               SEDT_QRY                               RCIS_P3.PRG
*                               WTAR_QRY                               RCIS_P3.PRG
*                               INDV_QRY                               RCIS_P3.PRG
*                               PAYI_QRY                               RCIS_P3.PRG
*
*-----*

```

```

PROCEDURE M_PROMPT

```

```

*
```

```

@ 21, 0

```

```

M_CHOICE = .T.

```

```

@ 21,16 SAY 'DO YOU WANT TO CONTINUE WITH THIS MODE [Y/N]? ';

```

```

    GET M_CHOICE PICTURE 'Y'

```

```

CLEAR TYPEAHEAD

```

```

READ

```

```

IF .NOT. M_CHOICE

```

```

    @ 21, 0

```

```

    @ 21,33 SAY 'CLOSING FILES'

```

```

    @ 24, 0

```

```

ENDIF

```

```

RETURN

```

```

*-----*
*                                RO_CHK                                *
*-----*
*
* SUMMARY:
*     The RO_CHK procedure is invoked to check the validity of the rela-
*     tional operators entered on the query input screen. Invalid en-
*     tries are flagged and passed back to the invoking procedure.
*
* INVOKING PROCEDURES:
*
*      Procedure Name      Location
*      -----
*      WPSS_QRY            RCIS_P3.PRG
*      SCHA_QRY            RCIS_P3.PRG
*      DCFY_QRY            RCIS_P3.PRG
*      CLAS_QRY            RCIS_P3.PRG
*      HRAX_QRY            RCIS_P3.PRG
*      CGDT_QRY            RCIS_P3.PRG
*      SEDT_QRY            RCIS_P3.PRG
*      WTAR_QRY            RCIS_P3.PRG
*      INDV_QRY            RCIS_P3.PRG
*      PAYI_QRY            RCIS_P3.PRG
*
*-----*

```

# ``` PROCEDURE RO_CHK ```

```
*
```

```
  PARAMETER ROCHK
```

```
*
```

```
  GOOD_RO = .F.
```

```
  DO CASE
```

```
    CASE ROCHK = '<>'
```

```
      GOOD_RO = .T.
```

```
    CASE (ROCHK = '=' ) .OR. (ROCHK = ' =')
```

```
      GOOD_RO = .T.
```

```
    CASE (ROCHK = '>' ) .OR. (ROCHK = ' >')
```

```
      GOOD_RO = .T.
```

```
    CASE (ROCHK = '<' ) .OR. (ROCHK = ' <')
```

```
      GOOD_RO = .T.
```

```
    CASE (ROCHK = '>=' ) .OR. (ROCHK = ' <=')
```

```
      GOOD_RO = .T.
```

```
  ENDCASE
```

```
*
```

```
  RETURN
```

```

*-----*
*                               SET_DB                               *
*-----*
*
* SUMMARY:
*       The SET_DB procedure is used to set up the string variables used
*       to identify the different source and destination database files
*       (both data and index files). All procedures in this file use
*       these strings (GLOBAL) as opposed to building their own.
*
* VARIABLE DECLARATIONS:
*
*       Variable Name      Status      Purpose
*       -----
*       S_PREFIX           LOCAL      Used to store a one letter identifier for
*                                     the source files.
*
*-----*

```

```

PROCEDURE SET_DBQ

```

```

*
  PRIVATE S_PREFIX
*
  M_FILE = 'X_CDT_MS'
  P_FILE = 'X_CDT_PY'
  CT_FILE = 'X_CDT_CT'

  * Designate code for access to active or inactive files. *

  IF (G_SELECT = 'H')
    S_PREFIX = 'A'
  ELSE
    S_PREFIX = 'I'
  ENDIF
  M_FILE = STUFF(M_FILE,1,1,LTRIM(S_PREFIX))
  P_FILE = STUFF(P_FILE,1,1,LTRIM(S_PREFIX))
  CT_FILE = STUFF(CT_FILE,1,1,LTRIM(S_PREFIX))
*
  M_NDX = 'X_XXXX'
  P_NDX = 'X_XXXX'
  CT_NDX = 'X_ASCL'

  * Build index string variables used to build query index files. *

  DO CASE
    CASE QS_SELECT = 'H'
      M_NDX = 'X_WPSS'
      M_NDX_STR = 'AS_CLASS+(WPSS/1000.0)'
    CASE QS_SELECT = 'I'
      M_NDX = 'X_SCHA'
      M_NDX_STR = 'AS_CLASS+(CUM_GPA/10.0)'
    CASE QS_SELECT = 'J'
      M_NDX = 'X_DCFY'
      M_NDX_STR = 'YEAR(COM_DATE+92)+(FY_RTNG/100.00)+(DC_RTNG/1000.000)'

```

```

CASE (QS_SELECT = 'K') .OR. (QS_SELECT = 'I') .OR. (QS_SELECT = 'O')
  M_NDX      = 'X_CLAS'
  M_NDX_STR  = 'STR(AS_CLASS,1)+CAT_TYPE+L_NAME+F_NAME'
CASE QS_SELECT = 'M'
  M_NDX      = 'X_CGDT'
  M_NDX_STR  = 'STR(AS_CLASS,1)+STR(YEAR(COM_DATE),4)';
              + 'STR(MONTH(COM_DATE),2)+STR(DAY(COM_DATE),2)'
CASE QS_SELECT = 'N'
  M_NDX      = 'X_SEDT'
  M_NDX_STR  = 'STR(AS_CLASS,1)+STR(YEAR(SCHLR_DATE),4)';
              + 'STR(MONTH(SCHLR_DATE),2)+STR(DAY(SCHLR_DATE),2)';
              + 'STR(SCHLR_TYPE,3,1)'
CASE QS_SELECT = 'P'
  M_NDX      = 'X_SSAN'
  M_NDX_STR  = 'SSAN'
CASE QS_SELECT = 'Q'
  M_NDX      = 'X_SSAN'
  P_NDX      = 'X_PAYD'
  M_NDX_STR  = 'SSAN'
  P_NDX_STR  = 'SSAN+STR(YEAR(PAY_DATE1),4)+STR(MONTH(PAY_DATE1),2)';
              + 'STR(DAY(PAY_DATE1),2)'
ENDCASE
M_NDX      = STUFF(M_NDX,1,1,LTRIM(S_PREFIX))
P_NDX      = STUFF(P_NDX,1,1,LTRIM(S_PREFIX))
CT_NDX     = STUFF(CT_NDX,1,1,LTRIM(S_PREFIX))
M_NDX_F    = M_NDX + '.NDX'
P_NDX_F    = P_NDX + '.NDX'
CT_NDX_F   = CT_NDX + '.NDX'
*
RETURN

```

```

*-----*
*               DB3_Q_ERR               *
*-----*
* SUMMARY:                               *
*   The DB3_Q_ERR procedure displays system error messages and provides *
*   limited corrective action capabilities.  If a corrupted index con- *
*   dition is detected, the system attempts to repair it by creating a *
*   replacement.  For other errors, the system will display an advisory *
*   message and the error number detected.  This error number can be *
*   used to locate the problem area.  An exact decoding of error num- *
*   bers can be found in the dBASE III PLUS User's Manual Appendices. *
*
* INVOKING PROCEDURES:                   *
*
*      Procedure Name                     Location
*      -----
*      QUERIES                           RCIS_P3.PRG
*      WPSS_QRY                           RCIS_P3.PRG
*      SCHA_QRY                           RCIS_P3.PRG
*      DCFY_QRY                           RCIS_P3.PRG
*      CLAS_QRY                           RCIS_P3.PRG
*      HRAX_QRY                           RCIS_P3.PRG
*      CGDT_QRY                           RCIS_P3.PRG
*      SEDT_QRY                           RCIS_P3.PRG
*      WTAR_QRY                           RCIS_P3.PRG
*      INDV_QRY                           RCIS_P3.PRG
*      PAYI_QRY                           RCIS_P3.PRG
*
* VARIABLE DECLARATIONS:
*
*      Variable Name      Status      Purpose
*      -----
*      ERR_NUM            PARAMETER    Used to hold the system error number
*                                     returned by the built-in function ERROR().
*
*      ERR_MSG            PARAMETER    Used to hold the system error number re-
*                                     turned by the built-in function MESSAGE().
*
*      PRFX_SAV           LOCAL        Used to store a one letter identifier for
*                                     the source files.
*
*-----*

```

```

PROCEDURE DB3_Q_ERR
*
  PARAMETERS ERR_NUM, ERR_MSG
*
  PRIVATE PRFX_SAV
*
  @ 21, 0
  ? CHR(7)
  @ 21, 0
  ? CHR(7)
  @ 21, 0

```

```

? CHR(7)

* If an index error has occurred, try to correct the error by reindexing *
* all query index files using appropriate index string variables.      *

IF (ERR_NUM = 68) .OR. (ERR_NUM = 114)
  @ 21, 0
  @ 21,15 SAY 'INDEX ERROR DETECTED.  ATTEMPTING TO REBUILD INDICES.'
  @ 24,0
  IF FILE(M_NDX_F)
    REINDEX ON &M_NDX_STR TO &M_NDX
  ENDIF
  IF (QS_SELECT = 'H' .OR. QS_SELECT = 'I' .OR. QS_SELECT = 'J' .OR. ;
      QS_SELECT = 'P')
*
    IF FILE(CT_NDX_F)
      REINDEX ON AS_CLASS TO &CT_NDX
    ENDIF
  ENDIF
*
  IF (QS_SELECT = 'O')
    IF FILE('T_HGHT.NDX')
      INDEX ON HEIGHT TO T_HGHT
    ENDIF
    IF FILE('T_AGEC.NDX')
      INDEX ON AGE_CAT TO T_AGEC
    ENDIF
  ENDIF
*
  IF (QS_SELECT = 'Q')
    IF FILE(P_NDX_F)
      REINDEX ON &P_NDX_STR TO &P_NDX
    ENDIF
  ENDIF
  @ 21, 0
  ? CHR(7)
  @ 21,15 SAY 'INDICES REBUILT.  ATTEMPTING TO CONTINUE PROCESSING.'
  @ 21, 0
  RETRY
ELSE
  IF (ERR_NUM = 126)
    @ 23, 0
    @ 23,10 SAY 'PRINTER ERROR. CHECK PRINTER AND PRESS ANY KEY TO' ;
              + ' CONTINUE.'
    CLEAR TYPEAHEAD
    WAIT ' '
    @ 23, 0
  ELSE
    @ 22, 0
    @ 23, 0
    @ 22, 0 SAY ERR_MSG
    @ 23, 0 SAY 'REPORT ERROR CODE ['
    @ 23,19 SAY ERR_NUM PICTURE '@B ###'
    @ 23,22 SAY '].  PRESS ANY KEY TO CONTINUE.'
    CLEAR TYPEAHEAD

```

```
    WAIT ' '  
    @ 22, 0  
    @ 23, 0  
ENDIF  
ENDIF  
*  
RETURN
```

```

*-----*
*               BEGINNING OF RCISUTIL.PRG               *
*-----*
*
* SUMMARY:
*   The RCISUTIL procedure is the main driver for the RCIS utilities
*   function. This module initializes program variables, activates a
*   pop-up menu to determine user processing requirements, and invokes
*   procedures to reload & backup database files and to change author-
*   ization password.
*
* CALLED PROCEDURES:
*
*   Procedure Name      Location
*   -----
*   INIT                RCISUTL2.PRG
*   MENU                MENU.BIN
*   UBACKUP             RCISUTL2.PRG
*   URELOAD             RCISUTL2.PRG
*   PASSWORD            RCISUTL2.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   U_PARA              GLOBAL      Parameter for MENU.BIN that passes pop-up
*                                   function menu descriptions and returns with
*                                   user selection. A more detailed discussion
*                                   of this parameter is provided in RCIS_P1.PRG
*
*-----*

```

```

PUBLIC U_PARA
*
SET STATUS OFF
SET SCOREBOARD OFF
*
@ 1, 0 TO 3,79
@ 2,32 SAY 'RCIS UTILITIES'
@ 5, 0 TO 17,79
@ 7,33 SAY 'Version 1.10'
@ 9,30 SAY 'Copyright (C) 1987'
@ 11,38 SAY 'by'
@ 13,31 SAY 'Carter L. Frank'
@ 15,30 SAY 'All rights reserved'
@ 23, 0

* Designate RCISUTL2.PRG as the active procedure file. *

SET PROCEDURE TO RCISUTL2

* Call procedure U_INIT from RCISUTL2.PRG *

DO U_INIT
@ 5,0 CLEAR

```



```

*
LOOP_CNTRL = .T.

* Continue loop until user selects the "Done" option.  *

DO WHILE (LOOP_CNTRL)
  U_PARA = STUFF(U_PARA,1,1,'A')

  * Call menu assembly routine, passing the utility menu parameter.  *

  CALL MENU WITH U_PARA
  F_SELECT = SUBSTR(U_PARA,6,1)
  DO CASE
    CASE F_SELECT = 'H'
      DO UBACKUP
    CASE F_SELECT = 'I'
      DO URELOAD
    CASE F_SELECT = 'J'
      DO PASSWORD
    CASE F_SELECT = 'K'
      EXIT
  ENDCASE
ENDDO

* Restore initial dBASE III PLUS environment.  *

SET CONFIRM OFF
SET SCOREBOARD ON
SET TALK ON
SET ESCAPE ON
SET SAFETY ON
SET BELL ON
SET STATUS ON
CLEAR ALL
*
RETURN

```

```

*-----*
*               BEGINNING OF RCISUTI.2.PRG               *
*-----*
*               U_INIT                                   *
*-----*
* SUMMARY:                                              *
*     U_INIT is the main initialization procedure for the RCIS utilities *
*     function. This module initializes the database and index file    *
*     string variables and builds the character string which is used by *
*     menu to build the pop-up menu.                                *
*-----*
* VARIABLE DECLARATIONS:                                *
*-----*
*     Variable Name      Status      Purpose            *
*     -----            -
*     NDX_STRG           GLOBAL      String variable which contains all possible*
*                                   database index file names.                *
*-----*
*     FIL_STRG           GLOBAL      String variable which contains all possible*
*                                   database data file names.                  *
*-----*
*     -----           LOCAL      All local variables are explicitly defined *
*                                   in the RCIS_P1.PRG program.                  *
*-----*

```

# PROCEDURE U\_INIT

```

*
* PUBLIC NDX_STRG
* PUBLIC FIL_STRG
*
* SET DELETED OFF
* SET CONFIRM ON
* SET CENTURY ON
* SET BELL OFF
* SET TALK OFF
* SET ESCAPE OFF
* SET SAFETY OFF
* LOAD MENU.BIN
*
* NDX_STR1 = "X_ASCL.NDX,X_CGDT.NDX,X_CLAS.NDX,X_DCFY.NDX,X_PAYD.NDX,X_SCHA.NDX"
* NDX_STR2 = ",X_SEDT.NDX,X_SSAN.NDX,X_WPSS.NDX,X_AGE.C.NDX,X_HGHT.NDX"
* NDX_STRG = NDX_STR1 + NDX_STR2
*
* FIL_STR1 = "X_CDT_CT.DBF,X_CDT_MS.DBF,X_CDT_PY.DBF,X_CDT_HW.DBF,X_CDT_RT.DBF,"
* FIL_STR2 = "X_CDT_WP.DBF"
* FIL_STRG = FIL_STR1 + FIL_STR2
*
* TL_BOX = CHR(201)
* X_BAR  = CHR(205) + CHR(205) + CHR(205) + CHR(205) + CHR(205)
* X_BAR  = X_BAR + X_BAR
* TR_BOX = CHR(187)

```

```

LM_BOX = CHR(204)
RM_BOX = CHR(185)
V_BAR  = CHR(186)
BL_BOX = CHR(200)
BR_BOX = CHR(188)
*
SEQ_1  = CHR(65 + 0)
ACT_1  = CHR(64 + 1)
SROW_1 = CHR(65 + 4)
SCOL_1 = CHR(65 + 34)
BROW_1 = CHR(65 + 11)
AROW_1 = CHR(65 + 7)
SLEN_1 = CHR(65 + 12)
*
U_PARA = SEQ_1 + ACT_1 + SROW_1 + SCOL_1 + BROW_1 + AROW_1 + SLEN_1
U_PARA = U_PARA + TL_BOX + X_BAR + TR_BOX
U_PARA = U_PARA + V_BAR + ' FUNCTION ' + V_BAR
U_PARA = U_PARA + LM_BOX + X_BAR + RM_BOX
U_PARA = U_PARA + V_BAR + ' BackUp   ' + V_BAR
U_PARA = U_PARA + V_BAR + ' ReLoad   ' + V_BAR
U_PARA = U_PARA + V_BAR + ' PassWord ' + V_BAR
U_PARA = U_PARA + V_BAR + ' Done     ' + V_BAR
U_PARA = U_PARA + BL_BOX + X_BAR + BR_BOX
*
RETURN

```

```

*-----*
*                                CHK_NDX                                *
*-----*
*
* SUMMARY:
*     The CHK_NDX procedure is used by the Reload function to erase any
*     existing database index files that are on the main disk drive (hard
*     disk drive labeled C).
*
* VARIABLE DECLARATIONS:
*
* Variable Name      Status      Purpose
*-----*
*     STRT_POS        LOCAL        Used as a pointer to locate the beginning
*                                   of each file name.
*
*     PRFX_LTR         LOCAL        Used to store a one letter identifier for
*                                   the active and inactive database files.
*
*     MAX_POS          LOCAL        Used to indicate different transition
*                                   points within the string variables.
*-----*

```

#### PROCEDURE CHK\_NDX

```

*
*
* STRT_POS = 1
* PRFX_LTR = 'A'
* MAX_POS = 99
* DO WHILE (PRFX_LTR <> 'X')
*     DO WHILE (STRT_POS < MAX_POS)
*         NDX_NAM_F = RTRIM(SUBSTR(NDX_STRG,STRT_POS,10))
*         NDX_NAM_F = STUFF(NDX_NAM_F,1,1,PRFX_LTR)
*         IF FILE(NDX_NAM_F)
*             ERASE &NDX_NAM_F
*         ENDIF
*         STRT_POS = STRT_POS + 11
*     ENDDO
*     IF (PRFX_LTR = 'A')
*         STRT_POS = 1
*         PRFX_LTR = 'I'
*     ELSE
*         IF (PRFX_LTR = 'I')
*             MAX_POS = 121
*             PRFX_LTR = 'T'
*         ELSE
*             PRFX_LTR = 'X'
*         ENDIF
*     ENDIF
* ENDDO
*
* RETURN

```



```

*-----*
*                                CHK_DSK                                *
*-----*
*
* SUMMARY:
*     The CHK_DSK procedure is used by the Backup function to erase any
*     existing database data files that are on the backup floppy disk
*     (disk drive labeled A).
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   STRT_POS           LOCAL      Used as a pointer to locate the beginning
*                                   of each file name.
*
*   PRFX_LTR           LOCAL      Used to store a one letter identifier for
*                                   the active and inactive database files.
*
*   MAX_POS            LOCAL      Used to indicate different transition
*                                   points within the string variables.
*-----*

```

PROCEDURE CHK\_DSK

```

*
  STRT_POS = 1
  PRFX_LTR = 'A'
  MAX_POS = 36
  DO WHILE (PRFX_LTR <> 'X')
    DO WHILE (STRT_POS < MAX_POS)
      FIL_NAM_F = SUBSTR(FIL_STRG,STRT_POS,12)
      FIL_NAM_F = 'A:' + STUFF(FIL_NAM_F,1,1,PRFX_LTR)
      IF FILE(FIL_NAM_F)
        ERASE &FIL_NAM_F
      ENDIF
      STRT_POS = STRT_POS + 13
    ENDDO
    IF (PRFX_LTR = 'A')
      STRT_POS = 1
      PRFX_LTR = 'I'
    ELSE
      IF (PRFX_LTR = 'I')
        MAX_POS = 72
        PRFX_LTR = 'T'
      ELSE
        PRFX_LTR = 'X'
      ENDIF
    ENDIF
  ENDDO
*
  RETURN

```

```

*-----*
*                               SET_DSK                               *
*-----*
*
* SUMMARY:
*   The SET_DSK procedure is used by the Reload function to erase any
*   data that exists on the database data files that are on the main
*   disk drive (hard disk drive labeled C).
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   STRT_POS           LOCAL      Used as a pointer to locate the beginning
*                                   of each file name.
*
*   PRFX_LTR           LOCAL      Used to store a one letter identifier for
*                                   the active and inactive database files.
*
*   MAX_POS            LOCAL      Used to indicate different transition
*                                   points within the string variables.
*-----*

```

PROCEDURE SET\_DSK

```

*
  SELECT 2
  STRT_POS = 1
  PRFX_LTR = 'A'
  MAX_POS = 36
  DO WHILE (PRFX_LTR <> 'X')
    DO WHILE (STRT_POS < MAX_POS)
      FIL_NAM_F = SUBSTR(FIL_STRG,STRT_POS,8)
      FIL_NAM_F = STUFF(FIL_NAM_F,1,1,PRFX_LTR)
      USE &FIL_NAM_F
      ZAP
      STRT_POS = STRT_POS + 13
    ENDDO
    IF (PRFX_LTR = 'A')
      STRT_POS = 1
      PRFX_LTR = 'I'
    ELSE
      IF (PRFX_LTR = 'I')
        MAX_POS = 72
        PRFX_LTR = 'T'
      ELSE
        PRFX_LTR = 'X'
      ENDIF
    ENDIF
  ENDDO
*
  USE
*
  RETURN

```





```

*-----*
*                                LOAD_DBF                                *
*-----*
*
* SUMMARY:
*     The LOAD_DBF procedure is used by the Reload function to copy data-
*     base data files from the floppy disk (disk drive labeled A) to the
*     main disk drive (hard disk drive labeled C).
*
* VARIABLE DECLARATIONS:
*
* Variable Name      Status      Purpose
*-----*-----*-----*
*     STRT_POS        LOCAL      Used as a pointer to locate the beginning
*                               of each file name.
*
*     PRFX_LTR         LOCAL      Used to store a one letter identifier for
*                               the active and inactive database files.
*
*     MAX_POS          LOCAL      Used to indicate different transition
*                               points within the string variable.
*-----*

```

#### PROCEDURE LOAD\_DBF

```

*
@ 20, 0
@ 20,14 SAY 'Insert backup diskette in drive A and press any key.'
CLEAR TYPEAHEAD
WAIT ' '
@ 20, 0
@ 20,21 SAY 'Loading backup files. Please wait...'
STRT_POS = 1
PRFX_LTR = 'A'
MAX_POS = 36
DO WHILE (PRFX_LTR <> 'X')
    DO WHILE (STRT_POS < MAX_POS)
        FIL_NAM_F = SUBSTR(FIL_STRG,STRT_POS,12)
        FIL_NAM_F = 'A:' + STUFF(FIL_NAM_F,1,1,PRFX_LTR)
        FIL_USE = SUBSTR(FIL_NAM_F,3,8)
        FIL_APND = SUBSTR(FIL_NAM_F,1,10)
        IF FILE(FIL_NAM_F)
            USE &FIL_USE
            APPEND FROM &FIL_APND
        ENDIF
        STRT_POS = STRT_POS + 13
    ENDDO
    IF (PRFX_LTR = 'A')
        STRT_POS = 1
        PRFX_LTR = 'I'
    ELSE
        IF (PRFX_LTR = 'I')
            MAX_POS = 72
            PRFX_LTR = 'T'

```

```
ELSE
  PRFX_LTR = 'X'
ENDIF
ENDIF
ENDDO
@ 20, 0
USE
*
RETURN
```

```

*-----*
*                                COPY_DBF                                *
*-----*
*
* SUMMARY:
*   The COPY_DBF procedure processes the database data files and puts
*   them on the backup floppy disk (disk drive labeled A). First, the
*   files are put into temporary files on the main disk (C) in a
*   specified sorted order. Next, they are copied to temporary files
*   on the floppy disk (A). As the system copies to the floppy disk
*   it continually checks the disk for the amount of available space.
*   If it runs out of space before the backup is finished, it prompts
*   the user to place another floppy disk in the disk drive. Once the
*   backup copying is complete the temporary files on the floppy disk
*   (A) are renamed with valid database file names and the temporary
*   sorted files on the main disk (C) are erased.
*
* CALLED PROCEDURES:
*
*                                Procedure Name      Location
*                                -----
*                                CHK_DSK              RCISUTL2.PRG
*
* VARIABLE DECLARATIONS:
*
* Variable Name      Status      Purpose
* -----
*   DBF_NAME          PARAMETER    String variable containing the complete
*                                   database file name to be processed.
*
*   TMP_SUB           PARAMETER    String variable containing a portion of the
*                                   database file name used to identify sort
*                                   fields.
*
*   PHASE             PARAMETER    Used to store a one letter identifier for
*                                   the active or the inactive database files.
*
*   SRT_NAME          LOCAL        String variable containing the name and
*                                   access path for the temporary sorted file
*                                   on the main drive (C).
*
*   TMP_NAME          LOCAL        String variable containing the name and
*                                   access path for the temporary file on the
*                                   floppy disk drive (A).
*
*   TARGET            LOCAL        String variable containing the name and
*                                   access path for the valid database file on
*                                   the floppy disk drive (A).
*
*   TMP_CNT           LOCAL        Used to store the number of records con-
*                                   tained in the database file being processed
*
*   REC_POS           LOCAL        Used to store the number of the current
*                                   database record being processed. Is com-
*                                   pared against TMP_CNT to ensure all
*                                   records have been processed.

```

```

*
*-----*
PROCEDURE COPY_DBF
*
  PARAMETERS DBF_NAME, TMP_SUB, PHASE
*
  SRT_NAME = 'C:S' + TMP_SUB + '.DBF'
  TARGET   = 'A:' + PHASE + TMP_SUB + '.DBF'
  TMP_NAME = 'A:X' + TMP_SUB + '.DBF'
  SELECT 1
  USE &DBF_NAME
  REC_CNT = RECCOUNT()
  COPY STRUCTURE TO &TMP_NAME
  IF REC_CNT > 0

    * If only one record, don't sort the file. *

    IF REC_CNT = 1
      COPY TO &SRT_NAME

    * If more than one record, sort the entire file. *

    ELSE
      DO CASE
        CASE TMP_SUB = '_CDT_MS'
          SORT TO &SRT_NAME ON SSAN
        CASE TMP_SUB = '_CDT_PY'
          SORT TO &SRT_NAME ON SSAN, PAY_DATE1
        CASE TMP_SUB = '_CDT_CT'
          SORT TO &SRT_NAME ON AS_CLASS
        CASE TMP_SUB = '_CDT_HW'
          SORT TO &SRT_NAME ON HEIGHT
        CASE TMP_SUB = '_CDT_RT'
          SORT TO &SRT_NAME ON AGE_CAT
      ENDCASE
    ENDIF
  ENDIF
  USE
  SELECT 2
  USE &TMP_NAME
  SET DEFAULT TO A:
  TMP_CNT = 0

  * Continue looping until all records have been processed. *

  DO WHILE (TMP_CNT < REC_CNT) .AND. (REC_CNT <> 0)

    * Copy from the sorted file until disk space runs low. *

    APPEND FROM &SRT_NAME FOR (DISKSPACE() > 10000)
    REC_POS = RECCOUNT()
    TMP_CNT = TMP_CNT + REC_POS
    IF REC_POS > 0

```

GO REC\_POS

\* Save the value of the sort field to be used as a starting \*  
\* point if the rest of the file needs to be put on another disk \*

```
DO CASE
  CASE TMP_SUB = '_CDT_MS'
    SSAN_VAL = SSAN
  CASE TMP_SUB = '_CDT_PY'
    SRTV1 = 'SSAN+STR(YEAR(PAY_DATE1),4)+STR(MONTH(PAY_DATE1),2)'
    SRTV2 = '+STR(DAY(PAY_DATE1),2)'
    SRT_VAL = SRTV1 + SRTV2
    PAY_VAL = &SRT_VAL
  CASE TMP_SUB = '_CDT_CT'
    ASCL_VAL = AS_CLASS
  CASE TMP_SUB = '_CDT_HW'
    HGHT_VAL = HEIGHT
  CASE TMP_SUB = '_CDT_RT'
    AGE_CAT = AGE_CAT
  AGE_CAT_VAL = AGE_CAT
```

ENDCASE

ENDIF

\* If the entire file did not fit on the same disk, prompt the \*  
\* user for another disk and delete that portion of the sort file \*  
\* already copied. \*

```
IF TMP_CNT < REC_CNT
  SELECT 2
  USE
  RENAME &TMP_NAME TO &TARGET
  @ 20, 0
  ? CHR(7)
  @ 20,14 SAY 'Insert a formatted disk in drive A and press any key.'
  CLEAR TYPEAHEAD
  WAIT ' '
  @ 20, 0
  SET DEFAULT TO C:
  @ 20,21 SAY 'Checking target disk. Please wait...'
  DO CHK_DSK
  @ 20, 0
  @ 20,20 SAY 'Continuing with backup. Please wait...'
  SELECT 1
  USE &SRT_NAME
  COPY STRUCTURE TO &TMP_NAME
  DO CASE
    CASE TMP_SUB = '_CDT_MS'
      DELETE FOR SSAN <= SSAN_VAL
    CASE TMP_SUB = '_CDT_PY'
      DELETE FOR &SRT_VAL <= PAY_VAL
    CASE TMP_SUB = '_CDT_CT'
      DELETE FOR AS_CLASS <= ASCL_VAL
    CASE TMP_SUB = '_CDT_HW'
      DELETE FOR HEIGHT <= HGHT_VAL
    CASE TMP_SUB = '_CDT_RT'
      DELETE FOR AGE_CAT <= AGE_CAT_VAL
```

```
        ENDCASE
        PACK
        USE
        SELECT 2
        USE &TMP_NAME
        SET DEFAULT TO A:
    ENDIF
ENDDO
SET DEFAULT TO C:
USE
RENAME &TMP_NAME TO &TARGET
IF FILE(SRT_NAME)
    ERASE &SRT_NAME
ENDIF
*
RETURN
```

```

*-----*
*                                UBACKUP                                *
*-----*
*
* SUMMARY:
*   The UBACKUP procedure is the main driver for the Backup function.
*   It sets up the string variables needed to process the backup and
*   invokes the appropriate procedures to process them.
*
* CALLED PROCEDURES:
*
*                                Procedure Name      Location
*                                -----
*                                CHK_DSK             RCISUTL2.PRG
*                                COPY_DBF           RCISUTL2.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   STRT_POS           LOCAL       Used as a pointer to locate the beginning
*                                   of each file name.
*
*   PRFX_LTR           LOCAL       Used to store a one letter identifier for
*                                   the active and inactive database files.
*
*   MAX_POS            LOCAL       Used to indicate different transition
*                                   points within the string variables.
*-----*

```

# PROCEDURE UBACKUP

```

*
@ 20, 0
@ 20,14 SAY 'Insert a formatted disk in drive A and press any key.'
CLEAR TYPEAHEAD
WAIT ' '
@ 20, 0
@ 20,21 SAY 'Checking target disk. Please wait...'
DO CHK_DSK
@ 20, 0
@ 20,21 SAY 'Starting RCIS backup. Please wait...'
STRT_POS = 1
PRFX_LTR = 'A'
MAX_POS = 36
DO WHILE (PRFX_LTR <> 'X')
  DO WHILE (STRT_POS < MAX_POS)
    DBF_F_NAM = SUBSTR(FIL_STRG,STRT_POS,8)
    DBF_F_NAM = 'C:' + STUFF(DBF_F_NAM,1,1,PRFX_LTR)
    TMP_NAM = SUBSTR(DBF_F_NAM,4,7)
    DO COPY_DBF WITH DBF_F_NAM, TMP_NAM, PRFX_LTR
    STRT_POS = STRT_POS + 13
  ENDDO
  IF (PRFX_LTR = 'A')
    STRT_POS = 1
  PRFX_LTR = 'X'

```

```

        PRFX_LTR = 'I'
    ELSE
        IF (PRFX_LTR = 'I')
            MAX_POS = 72
            PRFX_LTR = 'T'
        ELSE
            PRFX_LTR = 'X'
        ENDIF
    ENDIF
ENDDO
SELECT 1
USE
SELECT 2
USE
@ 20, 0
@ 20,18 SAY 'Backup complete.  Press any key to continue.'
CLEAR TYPEAHEAD
WAIT ' '
@ 20, 0
*
RETURN

```



```

*-----*
*                                URELOAD                                *
*-----*
* SUMMARY:                                                                *
*   The URELOAD procedure is the main driver for the Reload function.  *
*   It requires the user to input a password which is checked against  *
*   the system password for validity. It invokes procedures which pre- *
*   pare the system files for reload and prompts the user for the num- *
*   ber of reload disks to process.                                     *
*
* CALLED PROCEDURES:
*
*   Procedure Name      Location
*   -----
*   CHK_NDX             RCISUTL2.PRG
*   SET_DSK             RCISUTL2.PRG
*   LOAD_DBF            RCISUTL2.PRG
*
* VARIABLE DECLARATIONS:
*
*   Variable Name      Status      Purpose
*   -----
*   PWORD              LOCAL       Used to store the user input password which
*                                   is checked against the system password
*
*   DSK_NO             LOCAL       Used to store the user input for number of
*                                   disks to process for the reload.
*
*   CUR_DSK            LOCAL       Used to keep track of the current disk
*                                   being processed.
*-----*

```

# PROCEDURE URELOAD

```

*
OPTION = .F.
@ 20, 0
? CHR(7)
@ 20, 0 SAY 'WARNING: This option will erase existing files.'
@ 20, 0+
@ 20, 26 SAY 'Do you want to continue? ' GET OPTION PICTURE 'Y'
CLEAR TYE AHEAD
READ
@ 20, 0
@ 22, 0
IF OPTION
  SELECT 1
  USE RCIS_PW
  PWORD = ' '
  @ 20, 28 SAY 'Enter password ' GET PWORD PICTURE '!!!!!!!'
  CLEAR TYPE AHEAD
  READ
  @ 20, 0
  IF ACCESS_PW <> PWORD

```

```

? CHR(7)
@ 20,19 SAY 'Access denied. Press any key to continue.'
CLEAR TYPEAHEAD
WAIT ' '
@ 20, 0
ELSE
@ 20,22 SAY 'Erasing existing RCIS indices. Please wait...'
DO CHK_NDX
@ 20, 0
@ 20,23 SAY 'Erasing existing RCIS files. Please wait...'
DO SET_DSK
@ 20, 0
DSK_NO = 0
@ 20,22 SAY 'How many disks will be processed? ';
GET DSK_NO PICTURE '@Z ##'
CLEAR TYPEAHEAD
READ
@ 20, 0
IF DSK_NO > 0
CUR_DSK = 1
DO WHILE CUR_DSK <= DSK_NO
DO LOAD_DBF
CUR_DSK = CUR_DSK + 1
ENDDO
@ 20, 0
@ 20,17 SAY 'Reload complete. Press any key to continue.'
CLEAR TYPEAHEAD
WAIT ' '
ENDIF
ENDIF
@ 20, 0
SELECT 1
USE
ENDIF
*
RETURN

```

```

*-----*
*                                     PASSWORD                                     *
*-----*
*
* SUMMARY:                                                                    *
*     The PASSWORD procedure allows the user to change the system pass-      *
*     word. The user is required to know and input the current valid         *
*     password before the system will accept their new password.             *
*
* VARIABLE DECLARATIONS:                                                       *
*
* Variable Name      Status      Purpose                                     *
*-----*-----*-----*
*     OLDWORD        LOCAL      Used to store the user input password which  *
*                               is checked against the system password        *
*
*     NEWWORD        LOCAL      Used to store the user input for the new     *
*                               password they would like to use.             *
*
*     VERWORD        LOCAL      Used to store the user input which is com-   *
*                               pared against NEWWORD for system verifica-   *
*                               tion.                                         *
*-----*

```

# PROCEDURE PASSWORD

```

*
  OLDWORD = ' '
  @ 16,26 SAY 'Enter old password ' GET OLDWORD PICTURE '!!!!!!!'
  CLEAR TYPEAHEAD
  READ
  NEWWORD = ' '
  @ 18,26 SAY 'Enter new password ' GET NEWWORD PICTURE '!!!!!!!'
  CLEAR TYPEAHEAD
  READ
  VERWORD = ' '
  @ 20,26 SAY 'Verify new password ' GET VERWORD PICTURE '!!!!!!!'
  CLEAR TYPEAHEAD
  READ
  IF VERWORD = NEWWORD
    USE RCIS_PW
    IF OLDWORD = ACCESS_PW
      REPLACE ACCESS_PW WITH NEWWORD
      @ 22,17 SAY 'Password changed. Press any key to continue.'
      CLEAR TYPEAHEAD
      WAIT ' '
    ELSE
      ? CHR(7)
      @ 22,19 SAY 'Access denied. Press any key to continue.'
      CLEAR TYPEAHEAD
      WAIT ' '
    ENDIF
  ELSE
    ? CHR(7)
  ENDIF

```

```
@ 22,19 SAY 'Access denied. Press any key to continue.'  
CLEAR TYPEAHEAD  
WAIT ' '
```

```
ENDIF
```

```
@ 16, 0
```

```
@ 18, 0
```

```
@ 20, 0
```

```
@ 22, 0
```

```
*
```

```
RETURN
```

# MENU.ASM

## SUMMARY:

The MENU.ASM routine was written in assembler code by Stephen M. Curran. It accepts the menu parameters from the calling RCIS procedures and builds pop-up menus based on those parameters. It also provides the environment for the user to use the arrow keys to move a highlighted bar to the different menu options for them to make their selection. Once a selection has been made or an escape sequence has been executed, this routine passes a code back to the calling procedure which indicates how the user responded in the current menu.

```

TITLE      MENU.ASM
;
;          ORG      00H
;
CSEG       SEGMENT  BYTE PUBLIC 'PROG'
          ASSUME    CS:CSEG
;
; parameters passed by dBASE III PLUS:
;
; DS:[BX]    = menu sequence
; DS:[BX+1]  = active menu
; DS:[BX+2]  = start row
; DS:[BX+3]  = start column
; DS:[BX+4]  = bottom row
; DS:[BX+5]  = active row
; DS:[BX+6]  = string length
; DS:[BX+7]  = start of data strings
;
;*****
;*          MAIN ROUTINE          *
;*****
;
START      PROC      FAR
;
; save working registers to stack
;
          PUSH      AX
          PUSH      BX
          PUSH      CX
          PUSH      DX
          PUSH      DS
          PUSH      SS
          PUSH      SI
;
          MOV       AL,DS:[BX]
          CMP       AL,43H
          JNE       NEW_SCRN

```

```

        MOV     AX,41H
        MOV     DS:[BX],AL
        CALL    CUR_INIT
        CALL    VIDEO
        XOR     AX,AX
        MOV     AL,DS:[BX+2]
        ADD     AL,03H
        MOV     DS:[BX+5],AL
        JMP     ROW_MATRIX
NEW_SCRN: CALL    INIT
;
;  load SI with string index
;
        XOR     SI,SI
;
;  print menu labels
;
;*****
;*          LABEL: DO_BOX          *
;*****
;
DO_BOX:  PUSH    CX
        XOR     CH,CH
        MOV     CL,DS:[BX+6]
        SUB     CL,41H
DO_STR:  PUSH    CX
        PUSH    BX
;
;  get current video mode and page
;
;  on return: BH = video page
;
        XOR     AL,AL
        MOV     AH,0FH
        INT     10H
;
;  set cursor position
;
        XOR     AL,AL
        MOV     AH,02H
        INT     10H
        MOV     CX,BX
        POP     BX
        PUSH    BX
        MOV     AL,DS:[BX+7+SI]
        MOV     AH,09H
        MOV     BH,CH
        MOV     CX,01H
        MOV     BL,07H
        INT     10H
;
;  increment cursor column
;
        ADD     DL,01H
;

```

```

; increment string index
;
;         ADD     SI,01H
;
; decrement loop counter
;
;         POP     BX
;         POP     CX
;         LOOP    DO_STR
;         ADD     DH,01H
;         MOV     DL,DS:[BX+3]
;         SUB     DL,41H
;         POP     CX
;         LOOP    DO_BOX
ROW_MATRIX: CALL    CUR_INIT
;         CALL    VIDEO
;
; test if new sequence
;
;         MOV     AL,DS:[BX]
;         SUB     AL,41H
;         CMP     AL,01H
;         JNE     KEY_DB
;         JMP     EXIT
KEY_DB:    MOV     AH,00H
;         INT     16H
;         CMP     AL,00H
;         JE      DB_SPEC
;         JMP     DB_NOSP
DB_SPEC:   CMP     AH,50H
;         JNE     CUR_UP
;         CALL    CUR_INIT
;         CALL    VIDEO
;         MOV     AL,DS:[BX+4]
;         SUB     AL,42H
;         CMP     AL,DH
;         JNE     REV_VIDEO
;         MOV     DH,06H
REV_VIDEO: ADD     DH,01H
;         MOV     DL,DS:[BX+3]
;         SUB     DL,40H
;         XOR     CH,CH
;         MOV     CL,DS:[BX+6]
;         SUB     CL,43H
;         CALL    VIDEO
;         ADD     DH,41H
;         MOV     DS:[BX+5],DH
;         JMP     KEY_DB
CUR_UP:    CMP     AH,48H
;         JNE     KEY_DB
;         CALL    CUR_INIT
;         CALL    VIDEO
;         SUB     DH,01H
;         CMP     DH,06H
;         JNE     SET_VID

```

```

        MOV     DH, DS: [ BX+4 ]
        SUB     DH, 42H
SET_VID: MOV     DL, DS: [ BX+3 ]
        SUB     DL, 40H
        XOR     CH, CH
        MOV     CL, DS: [ BX+6 ]
        SUB     CL, 43H
        CALL    VIDEO
        ADD     DH, 41H
        MOV     DS: [ BX+5 ], DH
        JMP     KEY_DB
DB_NOSP: CMP     AH, 1CH
        JE      DB_ENTER
        MOV     AL, DS: [ BX+1 ]
        CMP     AL, 41H
        JNE     CHK_ESC
        JMP     KEY_DB
CHK_ESC: CMP     AH, 01H
        JE      ERASE
        JMP     KEY_DB
ERASE:   CALL    INIT
ERASE_BOX: PUSH   CX
        PUSH   BX
        PUSH   DX
        XOR     AH, AH
        MOV     AL, DS: [ BX+6 ]
        SUB     AL, 41H
        PUSH   AX
        XOR     AL, AL
        MOV     AH, 0FH
        INT     10H
        XOR     AL, AL
        MOV     AH, 02H
        INT     10H
        POP     AX
        MOV     CX, AX
        MOV     AH, 09H
        MOV     AL, 20H
        MOV     BL, 07H
        INT     10H
        POP     DX
        ADD     DH, 01H
        POP     BX
        POP     CX
        LOOP    ERASE_BOX
        MOV     AL, 41H
        MOV     DS: [ BX ], AL
        JMP     EXIT
DB_ENTER: MOV     AL, 42H
        MOV     DS: [ BX ], AL
;
; restore the original registers from
; the system stack
;
; *****

```



```

;*          LABEL: EXIT          *
;*****
;
EXIT:      POP      SI
          POP      SS
          POP      DS
          POP      DX
          POP      CX
          POP      BX
          POP      AX
          RET
START      ENDP
;
;*****
;*          SUBROUTINE: INIT      *
;*****
;
INIT       PROC      NEAR
;
;   get menu row count
;
;   load CL with the final menu row
;   subtract the initial menu row
;   increment row count
;
          XOR      CH,CH
          MOV      CL,DS:[BX+4]
          SUB      CL,DS:[BX+2]
          ADD      CL,01H
;
;   initialize cursor position registers
;
;   load DH with start row
;   convert from ASCII to integer value
;   load DL with start column
;   convert from ASCII to integer value
;
          MOV      DH,DS:[BX+2]
          SUB      DH,41H
          MOV      DL,DS:[BX+3]
          SUB      DL,41H
          RET
INIT       ENDP
;
;*****
;*          SUBROUTINE: VIDEO      *
;*****
;
VIDEO      PROC      NEAR
;
          PUSH     BX
CHG_VIDEO:  PUSH     CX
;
;   get current video mode and page
;

```

```

; on return BH = video page
;
;       XOR     AL,AL
;       MOV     AH,0FH
;       INT     10H
;
; set cursor position
;
;       XOR     AL,AL
;       MOV     AH,02H
;       INT     10H
;
; read character and attribute
;
; on return: AH = attribute
;           AL = character
;
;       XOR     AL,AL
;       MOV     AH,08H
;       INT     10H
;
; write reverse video of character
;
;       CMP     AH,70H
;       JNE     REVERSE
;       MOV     BL,07H
;       JMP     STRING
REVERSE: MOV     BL,70H
STRING:  MOV     CX,01H
;       MOV     AH,09H
;       INT     10H
;
; increment cursor column
;
;       ADD     DL,01H
;
; decrement loop counter
;
;       POP     CX
;       LOOP    CHG_VIDEO
;       PUSH    DX
;       MOV     DH,1AH
;       XOR     AX,AX
;       MOV     AH,02H
;       INT     10H
;       POP     DX
;       POP     BX
;       RET
VIDEO    ENDP
;
; *****
; *          SUBROUTINE: CUR_INIT          *
; *****
;
CUR_INIT PROC NEAR

```

```

; load CX with field length
;
; load CL with string length
; convert ASCII to integer value
; and adjust for border
;
        XOR     CH,CH
        MOV     CL,DS:[BX+6]
        SUB     CL,43H
;
; load DX with cursor position
;
; load DH with active row
; convert ASCII to integer value
; load DL with column
; convert ASCII to integer value
; and adjust for border window
;
        MOV     DH,DS:[BX+5]
        SUB     DH,41H
        MOV     DL,DS:[BX+3]
        SUB     DL,40H
        RET
CUR_INIT  ENDP
;
CSEG      ENDS
END

```

```

*-----*
*                               CDT_M.FMT                               *
*-----*
*
* SUMMARY:
*
*   The CDT_M format file contains the screen formats which allow the
*   user to make changes to the data items displayed on the screen.
*   There are four full screen pages in this format file.
*
*-----*

```

```

@ 1, 0 TO 3,79 DOUBLE
@ 2, 9 SAY 'INDIVIDUAL CADET DATA - PERSONAL INFORMATION' (Page 1 of 4)'
@ 4,11 SAY 'SSAN '
@ 4,17 SAY SSAN PICTURE '@R 999-99-9999'
@ 6, 6 SAY 'First Name' GET F_NAME PICTURE '!!!!!!!!!!!!!!'
@ 7, 5 SAY 'Middle Name' GET M_NAME PICTURE '!!!!!!!!!!!!!!'
@ 8, 7 SAY 'Last Name' GET L_NAME PICTURE '!!!!!!!!!!!!!!'
@ 4,46 SAY 'Matric #' GET MATRIC PICTURE '999999'
@ 6,45 SAY 'Birthdate' GET BIRTHDATE
@ 8,46 SAY 'Age' GET AGE PICTURE '99'
@ 8,56 SAY 'Sex' GET SEX PICTURE '!'
@ 11,37 SAY 'LOCAL '
@ 12, 0 TO 16,79
@ 10,36 TO 12,44
@ 13, 2 SAY 'Street Address' GET LOCAL_STRT
@ 14,12 SAY 'City' GET LOCAL_CITY
@ 15, 8 SAY 'Zip Code' GET LOCAL_ZIP PICTURE '@R 99999-NNNN'
@ 14,49 SAY 'Phone' GET LOCAL_PHON PICTURE '@R 999-9999'
@ 18,35 SAY 'PERMANENT '
@ 19, 0 TO 23,79
@ 17,34 TO 19,46
@ 20, 2 SAY 'Street Address' GET PERM_STRT
@ 21,12 SAY 'City' GET PERM_CITY
@ 22, 4 SAY 'State' GET PERM_STAT PICTURE 'AA'
@ 22,18 SAY 'Zip Code' GET PERM_ZIP PICTURE '@R 99999-NNNN'
@ 21,49 SAY 'Phone' GET PERM_PHON PICTURE '@R (999)999-9999'
READ
@ 19, 0 TO 23,79
@ 15, 0 TO 19,79
@ 9, 0 TO 15,79
@ 3, 0 TO 9,79
@ 1, 0 TO 3,79 DOUBLE
@ 2, 9 SAY 'INDIVIDUAL CADET DATA - ADMINISTRATIVE INFORMATION (Page 2 of 4)'
@ 4,24 SAY 'SSAN '
@ 4,30 SAY SSAN PICTURE '@R 999-99-9999'
@ 6,21 SAY 'AS Class' GET AS_CLASS RANGE 1,5
@ 8,16 SAY 'Category Type' GET CAT_TYPE PICTURE '!'
@ 4,47 SAY 'Four Year Cadet' GET FOUR_YR PICTURE 'Y'
@ 6,49 SAY 'Prior Service' GET PRIOR_SVC PICTURE 'Y'
@ 8,47 SAY 'Waiver Required' GET WAIVER_REQ PICTURE 'Y'
@ 10,11 SAY 'Semester Interview' GET SEM_INTRVW
@ 12,23 SAY 'Height' GET HEIGHT RANGE 58,83
@ 14,23 SAY 'Weight' GET WEIGHT

```

@ 10,52 SAY 'Weigh Date'	GET WEIGH_DATE	
@ 12,54 SAY 'Run Time'	GET RUN_TIME	PICTURE '@R 99:99'
@ 14,54 SAY 'Run Date'	GET RUN_DATE	
@ 16, 2 SAY 'Pursuing/Conditional Status'	GET PC_STATUS	PICTURE '!!'
@ 18,25 SAY 'Race'	GET RACE	PICTURE '!!'
@ 16,54 SAY 'FSP Date'	GET FSP_DATE	
@ 18,55 SAY 'Form 48'	GET FORM_48	
@ 20, 2 SAY 'Physical Qualification Date'	GET PHY_DATE	
@ 22,12 SAY 'Physical Category'	GET PHY_CAT	PICTURE '!!'
@ 20,47 SAY 'Graduation Date'	GET GRAD_DATE	
@ 22,47 SAY 'Commission Date'	GET COM_DATE	
READ		
@ 1, 0 TO 3,79 DOUBLE		
@ 2, 9 SAY 'INDIVIDUAL CADET DATA - ACADEMIC INFORMATION		(Page 3 of 4)'
@ 4, 0 SAY 'SSAN'		
@ 4, 5 SAY SSAN PICTURE '@R 999-99-9999'		
@ 4,19 SAY 'Scholarship Type'	GET SCHLR_TYPE	RANGE 0,4
@ 4,42 SAY 'Scholarship Expiration Date'	GET SCHLR_DATE	
@ 6, 0 SAY 'Major'	GET MAJOR	PICTURE '!!!!'
@ 6,14 SAY 'Semester GPA'	GET SEM_GPA	RANGE 0,4
@ 6,35 SAY 'Cumulative GPA'	GET CUM_GPA	RANGE 0,4
@ 6,59 SAY 'AFOQT Date'	GET AFOQT_DATE	
@ 8, 0 TO 18,37		
@ 8,40 TO 18,79		
@ 19, 0 TO 24,37		
@ 19,40 TO 24,79		
@ 8,56 SAY 'ACT SCORES'		
@ 8,12 SAY 'AFOQT SCORES'		
@ 19,14 SAY 'SAT SCORES'		
@ 19,44 SAY 'MINIMUM REQUIRED COURSES COMPLETE'		
@ 9, 9 SAY 'Quantitative'	GET AFOQT_QUAN	RANGE 0,99
@ 11,14 SAY 'Verbal'	GET AFOQT_VERB	RANGE 0,99
@ 13,03 SAY 'Academic Aptitude'	GET AFOQT_AA	RANGE 0,99
@ 15,15 SAY 'Pilot'	GET AFOQT_PLT	RANGE 0,99
@ 17,11 SAY 'Navigator'	GET AFOQT_NAV	RANGE 0,99
@ 9,56 SAY 'Math'	GET ACT_MATH	RANGE 0,36
@ 11,53 SAY 'English'	GET ACT_ENGL	RANGE 0,36
@ 13,45 SAY 'Natural Science'	GET ACT_NSCI	RANGE 0,36
@ 15,46 SAY 'Social Science'	GET ACT_SSCI	RANGE 0,36
@ 17,50 SAY 'Cumulative'	GET ACT_CUM	RANGE 0,36
@ 21, 7 SAY 'Math'	GET SAT_MATH	RANGE 0,800
@ 21,22 SAY 'Verbal'	GET SAT_VERB	RANGE 0,800
@ 23,12 SAY 'Cumulative'	GET SAT_CUM	RANGE 0,1600
@ 21,48 SAY 'Math'	GET M_R_MATH	PICTURE 'Y'
@ 21,62 SAY 'English'	GET M_R_ENGL	PICTURE 'Y'
@ 23,50 SAY 'Foreign Language'	GET M_R_FLAN	PICTURE 'Y'
READ		
@ 1, 0 TO 3,79 DOUBLE		
@ 2, 9 SAY 'INDIVIDUAL CADET DATA - CORPS INFORMATION		(Page 4 of 4)'
@ 4, 0 TO 10,79		
@ 5,21 SAY 'SSAN '		
@ 5,27 SAY SSAN		PICTURE '@R 999-99-9999'
@ 9, 2 SAY 'AS Class Rank'	GET AS_RNK_POS	
@ 9,20 SAY 'out of'		
@ 9,27 SAY CLAS_NUM		

@ 5,53 SAY 'Fiscal Year Rating'	GET FY_RTNG	RANGE 0,50
@ 7,42 SAY "Detachment Commander's Rating"	GET DC_RTNG	RANGE 0,8
@ 9,50 SAY 'Field Training Rating'	GET FT_RTNG	RANGE 0,999
@ 11, 0 TO 15,79		
@ 12,22 SAY 'ALTU'	GET ALTU	PICTURE 'Y'
@ 14, 2 SAY 'Field Training Completed'	GET FT_COMP	PICTURE 'Y'
@ 12,56 SAY "Pilot's License"	GET PLT_LICENS	PICTURE 'Y'
@ 14,57 SAY 'Part Time Work'	GET WORK	PICTURE 'Y'
@ 16, 0 TO 22,79		
@ 17,12 SAY 'Corps Position'	GET CORPS_POS	
@ 19, 9 SAY 'Corps Auxiliaries'		
@ 19,27	GET CORPS_AUX	PICTURE '@R !!!!!!!!!!!!!!!!!!!!!'
@ 21, 3 SAY 'Significant Information'	GET OTHER_INFO	

```

*-----*
*                               CDT_M_VU.FMT                               *
*-----*
*                               *
* SUMMARY:                               *
*   The CDT_M_VU format file contains the screen formats which only   *
*   allow the user to view data items displayed on the screen.  There *
*   are four full screen pages in this format file.                   *
*                               *
*-----*

```

```

@ 1, 0 TO 3,79 DOUBLE
@ 2, 9 SAY 'INDIVIDUAL CADET DATA - PERSONAL INFORMATION' (Page 1 of 4)'
@ 4,11 SAY 'SSAN '
@ 4,17 SAY 'SSAN' PICTURE '@R 999-99-9999'
@ 6, 6 SAY 'First Name'
@ 6,17 SAY 'F_NAME' PICTURE '!!!!!!!!!!!!!!'
@ 7, 5 SAY 'Middle Name'
@ 7,17 SAY 'M_NAME' PICTURE '!!!!!!!!!!!!!!'
@ 8, 7 SAY 'Last Name'
@ 8,17 SAY 'L_NAME' PICTURE '!!!!!!!!!!!!!!'
@ 4,46 SAY 'Matric #'
@ 4,55 SAY 'MATRIC' PICTURE '999999'
@ 6,46 SAY 'Age'
@ 6,50 SAY 'AGE' PICTURE '99'
@ 6,56 SAY 'Sex'
@ 6,60 SAY 'SEX' PICTURE '!'
@ 8,45 SAY 'Birthdate'
@ 8,55 SAY 'BIRTHDATE'
@ 11,37 SAY 'LOCAL '
@ 12, 0 TO 16,79
@ 10,36 TO 12,44
@ 13, 2 SAY 'Street Address'
@ 13,17 SAY 'LOCAL_STRT'
@ 14,12 SAY 'City'
@ 14,17 SAY 'LOCAL_CITY'
@ 15, 8 SAY 'Zip Code'
@ 15,17 SAY 'LOCAL_ZIP' PICTURE '@R 99999-NNNN'
@ 14,49 SAY 'Phone'
@ 14,55 SAY 'LOCAL_PHON' PICTURE '@R 999-9999'
@ 18,35 SAY 'PERMANENT '
@ 19, 0 TO 23,79
@ 17,34 TO 19,46
@ 20, 2 SAY 'Street Address'
@ 20,17 SAY 'PERM_STRT'
@ 21,12 SAY 'City'
@ 21,17 SAY 'PERM_CITY'
@ 22, 4 SAY 'State'
@ 22,10 SAY 'PERM_STAT'
@ 22,18 SAY 'Zip Code'
@ 22,27 SAY 'PERM_ZIP' PICTURE '@R 99999-NNNN'
@ 21,49 SAY 'Phone'
@ 21,55 SAY 'PERM_PHON' PICTURE '@R (999)999-9999'
READ

```

@ 19, 0 TO 23,79  
 @ 15, 0 TO 19,79  
 @ 9, 0 TO 15,79  
 @ 3, 0 TO 9,79  
 @ 1, 0 TO 3,79 DOUBLE  
 @ 2, 9 SAY 'INDIVIDUAL CADET DATA - ADMINISTRATIVE INFORMATION (Page 2 of 4)'  
 @ 4,24 SAY 'SSAN'  
 @ 4,30 SAY SSAN PICTURE '@R 999-99-9999'  
 @ 6,21 SAY 'AS Class'  
 @ 6,30 SAY AS\_CLASS  
 @ 8,16 SAY 'Category Type'  
 @ 8,30 SAY CAT\_TYPE PICTURE '!'  
 @ 4,47 SAY 'Four Year Cadet'  
 @ 4,63 SAY FOUR\_YR PICTURE 'Y'  
 @ 6,49 SAY 'Prior Service'  
 @ 6,63 SAY PRIOR\_SVC PICTURE 'Y'  
 @ 8,47 SAY 'Waiver Required'  
 @ 8,63 SAY WAIVER\_REQ PICTURE 'Y'  
 @ 10,23 SAY 'Height'  
 @ 10,30 SAY HEIGHT  
 @ 12,23 SAY 'Weight'  
 @ 12,30 SAY WEIGHT  
 @ 14,19 SAY 'Weigh Date'  
 @ 14,30 SAY WEIGH\_DATE  
 @ 10,44 SAY 'Semester Interview'  
 @ 10,63 SAY SEM\_INTRVW  
 @ 12,54 SAY 'Run Time'  
 @ 12,63 SAY RUN\_TIME PICTURE '@R 99:99'  
 @ 14,54 SAY 'Run Date'  
 @ 14,63 SAY RUN\_DATE  
 @ 16, 2 SAY 'Physical Qualification Date'  
 @ 16,30 SAY PHY\_DATE  
 @ 18,12 SAY 'Physical Category'  
 @ 18,30 SAY PHY\_CAT PICTURE '!'  
 @ 16,47 SAY 'Graduation Date'  
 @ 16,63 SAY GRAD\_DATE  
 @ 18,47 SAY 'Commission Date'  
 @ 18,63 SAY COM\_DATE  
 @ 20, 2 SAY 'Pursuing/Conditional Status'  
 @ 20,30 SAY PC\_STATUS PICTURE '!'  
 @ 22,25 SAY 'Race'  
 @ 22,30 SAY RACE PICTURE '!'  
 @ 20,54 SAY 'FSP Date'  
 @ 20,63 SAY FSP\_DATE  
 @ 22,55 SAY 'Form 48'  
 @ 22,63 SAY FORM\_48 PICTURE 'Y'  
 READ  
 @ 1, 0 TO 3,79 DOUBLE  
 @ 2, 9 SAY 'INDIVIDUAL CADET DATA - ACADEMIC INFORMATION (Page 3 of 4)'  
 @ 4, 0 SAY 'SSAN'  
 @ 4, 5 SAY SSAN PICTURE '@R 999-99-9999'  
 @ 4,19 SAY 'Scholarship Type'  
 @ 4,36 SAY SCHLR\_TYPE  
 @ 4,42 SAY 'Scholarship Expiration Date'  
 @ 4,70 SAY SCHLR\_DATE



@ 6, 0 SAY 'Major'	
@ 6, 6 SAY MAJOR	PICTURE '!!!!'
@ 6,13 SAY 'Cumulative GPA'	
@ 6,28 SAY CUM_GPA	
@ 6,35 SAY 'Semester GPA'	
@ 6,48 SAY SEM_GPA	
@ 6,59 SAY 'AFOQT Date'	
@ 6,70 SAY AFOQT_DATE	
@ 8, 0 TO 18,37	
@ 8,40 TO 18,79	
@ 19, 0 TO 24,37	
@ 19,40 TO 24,79	
@ 8,14 SAY 'ACT SCORES'	
@ 8,54 SAY 'AFOQT SCORES'	
@ 19,14 SAY 'SAT SCORES'	
@ 19,44 SAY 'MINIMUM REQUIRED COURSES COMPLETE'	
@ 9,14 SAY 'Math'	
@ 9,19 SAY ACT_MATH	
@ 11,11 SAY 'English'	
@ 11,19 SAY ACT_ENGL	
@ 13, 3 SAY 'Natural Science'	
@ 13,19 SAY ACT_NSCI	
@ 15, 4 SAY 'Social Science'	
@ 15,19 SAY ACT_SSCI	
@ 17, 8 SAY 'Cumulative'	
@ 17,19 SAY ACT_CUM	
@ 9,49 SAY 'Quantitative'	
@ 9,61 SAY AFOQT_QUAN	
@ 11,54 SAY 'Verbal'	
@ 11,61 SAY AFOQT_VERB	
@ 13,43 SAY 'Academic Aptitude'	
@ 13,61 SAY AFOQT_AA	
@ 15,55 SAY 'Pilot'	
@ 15,61 SAY AFOQT_PLT	
@ 17,51 SAY 'Navigator'	
@ 17,61 SAY AFOQT_NAV	
@ 21, 7 SAY 'Math'	
@ 21,12 SAY SAT_MATH	
@ 21,22 SAY 'Verbal'	
@ 21,29 SAY SAT_VERB	
@ 23,12 SAY 'Cumulative'	
@ 23,23 SAY SAT_CUM	
@ 21,48 SAY 'Math'	
@ 21,53 SAY M_R_MATH	PICTURE 'Y'
@ 21,62 SAY 'English'	
@ 21,70 SAY M_R_ENGL	PICTURE 'Y'
@ 23,50 SAY 'Foreign Language'	
@ 23,67 SAY M_R_FLAN	PICTURE 'Y'
READ	
@ 1, 0 TO 3,79 DOUBLE	
@ 2, 9 SAY 'INDIVIDUAL CADET DATA - CORPS INFORMATION	(Page 4 of 4)'
@ 4, 0 TO 10,79	
@ 5,21 SAY 'SSAN '	
@ 5,27 SAY SSAN	PICTURE '@R 999-99-9999'
@ 7,16 SAY 'WPSS Score'	

```

@ 7,27 SAY WPSS PICTURE '999.99'
@ 9, 2 SAY 'AS Class Rank'
@ 9,16 SAY AS_RNK_POS
@ 9,20 SAY 'out of'
@ 9,27 SAY CLAS_NUM
@ 5,53 SAY 'Fiscal Year Rating'
@ 5,72 SAY FY_RTNG
@ 7,42 SAY "Detachment Commander's Rating"
@ 7,72 SAY DC_RTNG
@ 9,50 SAY 'Field Training Rating'
@ 9,72 SAY FT_RTNG
@ 11, 0 TO 15,79
@ 12,22 SAY 'ALTU'
@ 12,27 SAY ALTU PICTURE 'Y'
@ 14, 2 SAY 'Field Training Completed'
@ 14,27 SAY FT_COMP PICTURE 'Y'
@ 12,56 SAY "Pilot's License"
@ 12,72 SAY PLT_LICENS PICTURE 'Y'
@ 14,57 SAY 'Part Time Work'
@ 14,72 SAY WORK PICTURE 'Y'
@ 16, 0 TO 22,79
@ 17,12 SAY 'Corps Position'
@ 17,27 SAY CORPS_POS
@ 19, 9 SAY 'Corps Auxiliaries'
@ 19,27 SAY CORPS_AUX PICTURE '@R !!!!!!'
@ 21, 3 SAY 'Significant Information'
@ 21,27 SAY OTHER_INFO

```